PEMBANGUNAN SISTEM *MARKETPLACE* YANG DAPAT MEREKOMENDASIKAN *GRUP FACEBOOK* YANG SESUAI DENGAN PRODUK MENGGUNAKAN *ALGORITME COSINE SIMILARITY*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer

Disusun oleh:
Ahmad Wildan Mukafi
NIM: 135150201111015

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PEMBANGUNAN SISTEM *MARKETPLACE* YANG DAPAT MEREKOMENDASIKAN *GRUP FACEBOOK* YANG SESUAI DENGAN PRODUK MENGGUNAKAN *ALGORITME COSINE SIMILARITY*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Ahmad Wildan Mukafi
NIM: 135150201111015

Skripsi ini telah diuji dan dinyatakan lulus pada
02 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Achmad Arwan, S.Kom, M.Kom	Denny Sagita R., S.Kom, M.Kom
NIP: 198408152008121004	NIP: 19851124 201504 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsurunsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 2 Agustus 2018

Ahmad Wildan Mukafi

NIM: 135150201111015

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas limpahan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "Pembangunan Sistem *Marketplace* Yang Dapat Merekomendasikan *Grup Facebook* Yang Sesuai Dengan Produk Menggunakan *Algoritme Cosine Similarity*".

Terselesainya penulisan skripsi ini dibantu oleh doa dan motivasi dari berbagai pihak. Untuk itu, penulis menyampaikan terima kasih kepada pihak-pihak berikut.

1. Bapak Achmad Arwan, S.Kom, M.Kom dan Bapak Denny Sagita Rusdianto, S.Kom, M.Kom selaku dosen pembimbing yang telah membimbing penulis dengan sabar dan memberikan saran yang baik demi terselesainya skripsi ini.
2. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D. selaku ketua jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
3. Segenap Bapak dan Ibu Dosen Program Studi Teknik Informatika yang telah membagikan ilmunya kepada penulis selama perkuliahan.
4. Kedua orang tua tercinta yang memberikan doa dan mencurahkan kasih sayangnya kepada penulis.
5. Semua pihak dan teman-teman lainnya yang ikut membantu dalam proses penyelesaian skripsi dan selalu memberikan semangat kepada penulis

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak kekurangan, sehingga saran dan kritik yang membangun sangat penulis harapkan. Semoga Allah SWT senantiasa melimpahkan rahmat dan karunia kepada seluruh pihak yang turut membantu penyelesaian skripsi ini. Semoga skripsi ini bermanfaat bagi semua pihak yang menggunakannya.

Malang, 2 Juli 2018

Penulis

ABSTRAK

Ahmad Wildan Mukafi, Pembangunan Sistem *Marketplace* Yang Dapat Merekomendasikan *Grup Facebook* Yang Sesuai Dengan Produk Menggunakan *Algoritme Cosine Similarity*

Pembimbing: Achmad Arwan, S.Kom, M.Kom dan Denny Sagita Rusdianto, S.Kom, M.Kom

Pesatnya perkembangan teknologi informasi membuat banyak manusia untuk menciptakan sebuah inovasi baru dalam hal teknologi. *Marketplace* adalah suatu bentuk dari hasil kemajuan teknologi saat ini. *Marketplace* sendiri adalah tempat atau wadah untuk melakukan pemasaran produk atau jasa memalui media internet. Seiring berjalannya waktu persaingan antar produk akan semakin meningkat, oleh karena itu dibutuhkanlah suatu cara pemasaran yang dapat menjangkau lebih banyak target pembeli yang susuai dengan produk. Pada saat ini *facebook* merupakan media sosial yang cukup banyak digunakan. Pada *facebook* terdapat fitur grup yang dinilai mampu untuk membantu memasarkan produk sesuai dengan target produk. Untuk memecahkan masalah di atas dibutuhkan sistem *marketplace* yang sudah terintegrasi dengan *facebook*, dengan memanfaatkan *algoritme cosine similarity* maka produk akan melihat tingkat kemiripan dari grup yang tersedia. Hasil dari analisi kebutuhan yang telah dilakukan didapatkan 41 kebutuhan fungsional dan 1 kebutuhan non fungsional. Pengujian yang dilakukan dalam penelitian ini adalah pengujian *whitebox* dan *blackbox*. Pada pengujian *whitebox* dari delapan operasi yang berbeda semuanya mendapatkan hasil 100% *valid* dan sesuai dengan analisis kebutuhan yang telah dibuat. Dan pada pengujian *blackbox* pada empat belas pengujian dengan metode *Equivalence Partitioning* hasilnya 100% *valid*. Dengan demikian hasil pengujian fungsionalitas dapat dikatakan sangat baik dan sudah sesuai dengan analisis kebutuhan.

Kata kunci: *marketplace, facebook, algoritme cosine similarity, whitebox, blackbox, Equivalence Partitioning*

ABSTRACT

Ahmad Wildan Mukafi, Pembangunan Sistem *Marketplace* Yang Dapat Merekomendasikan *Grup Facebook* Yang Sesuai Dengan Produk Menggunakan *Algoritme Cosine Similarity*

Pembimbing: Achmad Arwan, S.Kom, M.Kom dan Denny Sagita Rusdianto, S.Kom, M.Kom

The rapid development of information technology makes humans creating a new innovation in terms of technology. Marketplace is a result of the current technological advances. Marketplace is a place or a container to do product marketing or service through internet media. Over time the competition between products will increase, therefore it needs an advertising trick which can reach more target buyers who are appropriate towards the products. Nowadays, Facebook's being a social media which is pretty much used. Facebook has features which assessed the grup that be able to help advertising the product in accordance with the target of the product. To solve the problem above, a marketplace system which is already integrated with Facebook is required, by using cosine similarity algorithm then products will comprehend the level of similarity from the grup that have been existed. The results of the need analysis has been done, obtained fourty oe of the functional requirements and one non functional requirements. The test which is conducted in this research is the white box and black box testing. White box testing on eight different operating all getting the 100% valid and in accordance with the requirement analysis has been made. While black box testing that has been done on the fourteen testing with the Equivalence Partitioning method the results are 100% valid. As such the functionality test result can be said to be very good and compliance with the essential analysis

***Keywords**: marketplace, facebook, cosine similarity, whitebox, blackbox, Equivalence Partitioning*

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
ABSTRAK v
ABSTRACT vi
DAFTAR ISI vii
DAFTAR TABEL x
DAFTAR GAMBAR xiv
BAB 1 PENDAHULUAN 1
1.1 Latar belakang 1
1.2 Rumusan masalah 2
1.3 Tujuan 3
1.4 Manfaat 3
1.5 Batasan masalah 3
1.6 Sistematika pembahasan 3
BAB 2 LANDASAN KEPUSTAKAAN 5
2.1 Kajian Pustaka 5
2.2 Waterfall Software Development 5
2.3 *UML* 6
2.3.1 Use Case 7
2.3.2 Sequence Diagram 8
2.4 Pengujian Perangkat Lunak 10
2.4.1 Pengujian Fungsional 10
2.5 Algoritme Cosine Similarity 11
2.6 Marketplace 12
2.7 Facebook Graph API 12
2.8 Codeigniter 14
2.9 Boostrap 15
BAB 3 METODOLOGI 16
3.1 Studi Pustaka 16
3.2 Analisis Kebutuhan 16

3.3 Perancangan Sistem 17
3.4 Implementasi Sistem 17
3.5 Pengujian sistem 17
3.6 Kesimpulan 18
BAB 4 ANALISIS KEBUTUHAN DAN PERANCANGAN 19
4.1 Elisitasi kebutuhan 19
4.2 Analisis kebutuhan 23
4.2.1 Identifikasi aktor 23
4.2.2 Kebutuhan Fungsional Sistem 24
4.2.3 Kebutuhan Non-Fungsional Sistem 39
4.3 Permodelan Analisis Kebutuhan 40
4.3.1 *Use Case* Diagram 40
4.3.2 Skenario Use case 41
4.4 Perancangan Sistem 65
4.4.1 *Sequence Diagram* 65
4.4.2 Class Diagram 69
4.4.3 Perancangan *Algoritme* 72
4.4.4 Perancangan Antarmuka 78
BAB 5 IMPLEMENTASI SISTEM 84
5.1 Spesifikasi Sistem 84
5.1.1 *Spesifikasi Hardware* 84
5.1.2 *Spesifikasi Software* 84
5.2 Implementasi Sistem *Marketplace* 85
5.2.1 Implementasi Algoritme 85
5.2.2 Implementasi Antarmuka 95
5.2.3 Implementasi Basis data 99
BAB 6 PENGUJIAN 101
6.1 Pengujian *White-box* 101
6.1.1 Pengujian unit 101
6.1.2 Pengujian Integrasi 119
6.2 Pengujian *Black-box* 127
6.2.1 Pengujian Validasi Masuk admin 127

6.2.2 Pengujian Validasi Tambah Data Produk Kategori 128
6.2.3 Pengujian Validasi Perbarui Data Produk Kategori 129
6.2.4 Pengujian Validasi Tambah User Admin 129
6.2.5 Pengujian Validasi Daftar 130
6.2.6 Pengujian Validasi Masuk 131
6.2.7 Pengujian Validasi Tambah Produk 131
6.2.8 Pengujian Validasi Perbarui Produk 132
6.2.9 Pengujian Validasi Bagikan Info Produk 133
6.2.10 Pengujian Validasi Perbarui Profil Member 134
6.2.11 Pengujian Validasi Tambah item Keranjang Belanja 134
6.2.12 Pengujian Validasi Checkout 135
6.2.13 Pengujian Validasi Tambah Alamat 135
6.2.14 Pengujian Validasi Tambah Testimoni 136
6.2.15 Pengujian Compatibility 137
6.3 Pengujian Kebutuhan 138
6.3.1 Traceability Matrix 138
6.4 Analisis Pengujian 139
BAB 7 PENUTUP 141
7.1 Kesimpulan 141
7.2 Saran 141
DAFTAR PUSTAKA 143

DAFTAR TABEL

Tabel 2.1 Simbol *Use Case Diagram* 7
Tabel 2.2 Penjelasan simbol *Sequence Diagram* 8
Tabel 2.3 *Equivalence Partitioning* 11
Tabel 4.1 Fitur yang didapat dari hasil obeservasi 23
Tabel 4.2 Identifikasi aktor 24
Tabel 4.3 Tabel kebutuhan Fungsional 24
Tabel 4.4 Tabel kebutuhan non-fungsional 40
Tabel 4.5 *Skenario Use case* masuk admin 41
Tabel 4.6 Skenario *Use case* tampil *data* member 42
Tabel 4.7 Skenario *Use case* perbarui *status member* 42
Tabel 4.8 Skenario *Use case* tampil daftar produk *admin* 43
Tabel 4.9 Skenario *Use case* perbarui *status* produk 43
Tabel 4.10 Skenario *Use case* tampil *data* produk kategori 44
Tabel 4.11 Skenario *Use case* tambah tambah data kategori 44
Tabel 4.12 Skenario *Use case* perbarui data kategori produk 45
Tabel 4.13 Skenario *Use case* konfirmasi admin 45
Tabel 4.14 Skenario *Use case* tampil data *grup facebook* 46
Tabel 4.15 Skenario *Use case* tampil daftar data user admin 47
Tabel 4.16 Skenario *Use case* tambah *user admin* 47
Tabel 4.17 Skenario *Use case* perbarui *data user* admin 48
Tabel 4.18 Skenario *Use case register* 48
Tabel 4.19 Skenario *Use case* masuk 49
Tabel 4.20 Skenario *Use case* melihat katalog produk 49
Tabel 4.21 *Skenario Use case* lihat detail produk 50
Tabel 4.22 Skenario *Use case* Lihat daftar produk kategori 50
Tabel 4.23 Skenario *Use case* Pencarian Nama Produk 51
Tabel 4.24 Skenario *Use case* tambah produk 51
Tabel 4.25 Skenario *Use case* *perbarui* Produk 52
Tabel 4.26 Skenario *Use case* hapus Produk 52
Tabel 4.27 Skenario *Use case* Tampilkan Daftar produk penjual 53
Tabel 4.28 Skenario *Use case* Bagikan *info* Produk 53

Tabel 4.29 Skenario *Use case* Tambah *Grup Facebook* 55
Tabel 4.30 Skenario *Use case* Hapus *grup facebook* 55
Tabel 4.31 Skenario *Use case* Konfirmasi transaksi produk Penjual 56
Tabel 4.32 Skenario *Use case* Tampil *Profile* member 56
Tabel 4.33 Skenario *Use case* Perbarui *Profile* member 56
Tabel 4.34 Skenario *Use case* Tambah *Item* Keranjang belanja 57
Tabel 4.35 Skenario *Use case* Tampilkan keranjang belanja 58
Tabel 4.36 Skenario *Use case* Hapus Item Keranjang belanja 58
Tabel 4.37 Skenario *Use case* Transaksi jual 59
Tabel 4.38 Skenario *Use case* Riwayat belanja 59
Tabel 4.39 Skenario *Use case koneksi facebook* 60
Tabel 4.40 Skenario *Use case Similarity grup facebook* 61
Tabel 4.41 Skenario *Use case checkout* 62
Tabel 4.42 Skenario *Use case* Tampil daftar alamat member 62
Tabel 4.43 Skenario *Use case* Tambah alamat 63
Tabel 4.44 Skenario *Use case* Hapus alamat member 63
Tabel 4.45 Skenario *Use case* Tambah Testimoni 64
Tabel 4.46 *Pseudocode* fungsi koneksi *facebook* 72
Tabel 4.47 *Pseudocode* fungsi *cosine similarity* 73
Tabel 4.48 *Pseudocode* fungsi *post facebook* 74
Tabel 4.49 *Pseudocode* fungsi *addcart* 74
Tabel 4.50 *Pseudocode* fungsi *checkout* 75
Tabel 4.51 *Pseudocode* fungsi dapatkan kode tagihan pembayaran 76
Tabel 4.52 *Pseudocode* fungsi konfirmasi transaksi produk admin 76
Tabel 4.53 *Pseudocode* fungsi konfirmasi transaksi produk penjual 77
Tabel 5.1 *Spesifikasi Hardware* 84
Tabel 5.2 *Spesifikasi Software* 84
Tabel 5.3 Implementasi algoritme *connect facebook* 85
Tabel 5.4 Implementasi fungsi *cosine similarity* 86
Tabel 5.5 Implementasi Algoritme *post facebook* 89
Tabel 5.6 Implementasi Tambah Item Keranjang Belanja 91
Tabel 5.7 Implementasi Fungsi Checkout 92

Tabel 5.8 Implementasi Algoritme konfirmasi transaksi produk Admin 93
Tabel 5.9 Implementasi Algoritme konfirmasi penjual 94
Tabel 6.1 Pengujian Unit koneksi *Facebook* 101
Tabel 6.2 Hasil Pengujian unit *operasi index*() 103
Tabel 6.3 Pengujian Unit *cosine similarity* 104
Tabel 6.4 Hasil Pengujian unit operasi perhitungan() 106
Tabel 6.5 Pengujian Unit *Post facebook* 107
Tabel 6.6 Hasil Pengujian unit operasi postfacebook() 108
Tabel 6.7 Pengujian unit Addcart 108
Tabel 6.8 Hasil Pengujian unit operasi *Addcart*() 109
Tabel 6.9 Pengujian Unit checkout 110
Tabel 6.10 Hasil Pengujian unit operasi *checkout()* 112
Tabel 6.11 Pengujian unit dapatkan kode tagihan 112
Tabel 6.12 Hasil Pengujian unit operasi index() 114
Tabel 6.13 Pengujian Unit konfirmasi transaksi produk admin 115
Tabel 6.14 Hasil Pengujian unit operasi *update*() 117
Tabel 6.15 Pengujian Unit konfirmasi transaksi produk penjual 117
Tabel 6.16 Hasil Pengujian unit operasi update() 119
Tabel 6.17 Pembentukan node algoritme logoutfb 120
Tabel 6.18 Hasil Pengujian integrasi algoritme getListProduk 120
Tabel 6.19 Pembetukan *node* algoritme perhitungan 121
Tabel 6.20 Hasil Pengujian unit operasi perhitungan() 123
Tabel 6.21 Pembetukan *node* algoritme post 124
Tabel 6.22 Hasil Pengujian unit operasi *post*() 125
Tabel 6.23 Pembetukan *node* algoritme *Addcart* 126
Tabel 6.24 Hasil Pengujian unit operasi *Addcart*() 127
Tabel 6.25 Kasus Uji Validasi Masuk admin 127
Tabel 6.26 Kasus Uji Validasi Tambah Data Produk Kategori 128
Tabel 6.27 Kasus Uji Validasi Perbarui data Produk kategori 129
Tabel 6.28 Kasus Uji Validasi Tambah User Admin 129
Tabel 6.29 Kasus Uji Validasi Daftar 130
Tabel 6.30 Kasus Uji Validasi Masuk 131

Tabel 6.31 Kasus Uji Validasi Tambah Produk 131
Tabel 6.32 Kasus Uji Validasi Perbarui Produk..... 132
Tabel 6.33 Kasus Uji Validasi Bagikan Info Produk 133
Tabel 6.34 Kasus Uji Validasi Perbarui Profile Member..... 134
Tabel 6.35 Kasus Uji Validasi Tambah item Keranjang Belanja 134
Tabel 6.36 Kasus Uji Validasi checkout 135
Tabel 6.37 Kasus Uji Validasi Tambah Alamat 135
Tabel 6.38 Kasus Uji Validasi Tambah Testimoni..... 136
Tabel 6.39 Kasus Uji Menjalankan Fungsionalitas Sistem Pada Browser Yang Berbeda 137
Tabel 6.40 Hasil Pengujian Compatibility..... 138
Tabel 6.41 Traceability Matrix 138

DAFTAR GAMBAR

Gambar 2.1 Metode *Waterfall (Sommervile, 2011:30)* 6

Gambar 2.2 Contoh *Use Case Diagram (Pressman, 2010:847)* 7

Gambar 2.3 Contoh *Sequence Diagram* 8

Gambar 2.4 *Dominasi Facebook* tahun 2013 *(Lukman, 2013)* 13

Gambar 2.5 Alur dari *Facebook graph* (Kaur,2014) 13

Gambar 2.6 *Output Graph API* (kaur, 2014) 14

Gambar 2.7 *Alur Graph API* 14

Gambar 2.8 *Grid system Boostrap* (Sumber: coderomeos.org) 15

Gambar 3.1 Metodologi Penelitian 16

Gambar 4.1 Halaman utama pada sistem bukalapak (sumber: www.bukalapak.com) 19

Gambar 4.2 Halaman Detail produk sistem bukalapak (sumber: www.bukalapak.com) 20

Gambar 4.3 Halaman Keranjang belanja (sumber: www.bukalapak.com) 20

Gambar 4.4 Halaman transaksi dari sisi pembeli (sumber: www.bukalapak.com). 21

Gambar 4.5 Halaman pengaturan akun (sumber: www.bukalapak.com). 21

Gambar 4.6 Halaman tambah produk yang dijual (sumber: www.bukalapak.com) 22

Gambar 4.7 *Use Case Diagram* 40

Gambar 4.8 *Sequence diagram* keranjang belanja 65

Gambar 4.9 *Sequence diagram* lihat keranjang belanja 65

Gambar 4.10 *Sequence diagram Checkout* 66

Gambar 4.11 *Sequence diagram* Konfirmasi transaksi produk Penjual 67

Gambar 4.12 *Sequence diagram* konfirmasi transaksi produk admin 67

Gambar 4.13 *Sequence diagram* Similarity grup Facebook 68

Gambar 4.14 *Class diagram* 69

Gambar 4.15 *Class Diagram package model* 70

Gambar 4.16 *Class Diagram package controller* 71

Gambar 4.17 Perancangan antarmuka halaman Utama atau beranda 78

Gambar 4.18 Peracangan antarmuka halaman *detail* produk 79

Gambar 4.19 Perancangan antarmuka halaman koneksi facebook 80

Gambar 4.20 Perancangan antarmuka halaman bagikan *facebok* 80
Gambar 4.21 Perancangan antarmuka halaman keranjang belanja 81
Gambar 4.22 Perancangan antarmuka halaman checkout 81
Gambar 4.23 Perancangan antarmuka halaman konfirmasi produk admin 82
Gambar 4.24 Perancangan antarmuka konfirmasi produk penjual..... 82
Gambar 5.1 Informasi peninjauan graph API versi 3(Sumber: https://developers.facebook.com) 89
Gambar 5.2 Akses Graph API versi 2.9 (Sumber: https://developers.facebook.com) 90
Gambar 5.3 Akses Graph API versi 3 (Sumber: https://developers.facebook.com) 91
Gambar 5.4 Halaman Utama 95
Gambar 5.5 Halaman detail produk dan tambah item keranjang belanja 96
Gambar 5.6 Halaman Koneksi *facebook* 96
Gambar 5.7 Halaman bagikan grup facebook..... 97
Gambar 5.8 Halaman keranjang belanja..... 97
Gambar 5.9 Halaman *checkout* 98
Gambar 5.10 Halaman Konfirmasi transaksi produk Admin 98
Gambar 5.11 Halaman Konfirmasi transaksi produk Penjual 99
Gambar 5.12 Implementasi Basis Data..... 100
Gambar 6.1 *Flow Graph Operasi index()* 102
Gambar 6.2 *Floiw Graph Operasi* perhitungan()..... 105
Gambar 6.3 *Floiw Graph Operasi postFacebook*() 107
Gambar 6.4 *Flow graph* Operasi addcart..... 109
Gambar 6.5 *Flow Graph* Operasi checkout..... 111
Gambar 6.6 *Flow Graph Operasi getInvoice*()..... 113
Gambar 6.7 *Flow graph operasi update()* 116
Gambar 6.8 *Flow Graph Operasi update()* 118
Gambar 6.9 *Flow Graph* algoritme getListProduk 120
Gambar 6.10 *Floiw Graph Operasi* perhitungan()..... 122
Gambar 6.11 *Flow graph* Operasi *post* 124
Gambar 6.12 *Flow graph* Operasi addcart..... 126

BAB 1 PENDAHULUAN

1.1 Latar belakang

Pesatnya perkembangan teknologi informasi membuat banyak manusia untuk untuk menciptakan sebuah inovasi baru dalam hal teknologi. Dari permasalahan kecil hingga besar semua berlomba untuk menyelesaikan dalam bentuk teknologi. *Marketplace* adalah suatu bentuk dari hasil kemajuan teknologi saat ini. Dimana *marketplace* merupakan suatu bentuk sistem yang saling terintregasi satu dengan lainya. *Marketplace* sendiri bisa dikatakan sebagai salah satu terobosan yang bagus. Sehingga dapat disimpulkan bahwa dengan berkembangnya teknologi informasi dan sitem informasi, banyak permasalahan manusia di berbagai aspek kehidupan terselesaikan (Arifin, 2014).

Marketplace adalah tempat atau wadah untuk melakukan pemasaran produk atau jasa melalui media internet (wisnu, 2013). *Marketplace* populer saat ini seperti contoh Tokopedia, Bukalapak dan lain sebagainya. Sistem dari *marketplace* ini yang akan mempertemukan penjual dan pembeli, pihak *marketplace* itu sendiri tidak akan menyediakan barang atau produk dari mereka sendiri, tetapi terdapat penjual yang memasarkan produk atau barang mereka melalui sistem yang telah disediakan oleh *marketplace*. Pihak penjual ini akan mengunggah detail dari produk yang ingin dijual, baik itu nama produk, ukuran, warna, harga dan lain sebaginya. Disisi lain terdapat terdapat juga seorang pembeli yang akan melakuan pembelian barang atau produk dari penjual yang telah dipasarkan melalui sistem *marketplace*. Sehingga aktifitas dari *marketplace* itu sendiri meliputi pembeli melakukan pembelian barang melalui sistem *marketplace* yang telah disediakan, kemudian sistem *marketplace* akan memberitahu penjual bahwa produk mereka telah terjual melalui sistem *marketplace* tersebut. Setelah itu penjual akan merespon sesuai dengan pemberitahuan dari sistem dan melakukan pengiriman barang sesuai pesanan dari pembeli.

Facebook adalah suatu bentuk media berinteraksi sosial dalam bentuk media *online*, dan pada saat ini *facebook* merupakan media sosial yang cukup banyak digunakan oleh orang saat ini. Dari sekian banyak fitur terdapat satu fitur yang diharapkan bisa membantu dalam memasarkan produk, fitur tersebut adalah fitur grup *facebook.* Grup *facebook* adalah sekumpulan pengguna *Facebook* yang melakukan suatu pembahasan pada topik yang sama. Fitur grup tersebut kemudian digunakan untuk melakukan promosi produk-produk yang dinginkan oleh penjual. Seperti contoh seorang penjual melakukan promosi suatu barang, dimana orang tersebut akan melakukan pemasaran terhadap grup *facebook* yang dia ikuti. Untuk mendapatkan pembeli yang sesuai target, maka penjual akan melakukan *posting* produk pada setiap grup *facebook* yang sesuai dengan jenis produk tersebut. Dari hasil *posting* produk pada grup *facebook* diharapkan mampu menarik minat pembeli yang lebih banyak.

Pada saat ini *Facebook* menyediakan sebuah *API* dengan nama *Graph API* . *Graph API* adalah suatu layanan *facebook* untuk melakukan pemanggilan atau

pengiriman data oleh *user facebook,* dan juga dapat digunakan untuk mengirim dan mengambil data dalam proses penggunaan *social plugin*. Data user yang dapat dilakukan pemanggilan adalah *data* user yang sudah mendapat ijin hak akses oleh *user* tersebut. Dengan adanya *Graph API* pemanggilan *query* bisa disederhanakan dalam bentuk *path* sehingga hal tersebut mempermudah *developer* dalam memanggilnya (Limpo, 2012).

Algoritme *Cosine Similarity* merupakan algoritme yang digunakan untuk mengukur seberapa besar tingkat kemiripan dari dua buah *data string*. Pada penelitian Herwijayanti yang berjudul "*klasifikasi berita online dengan menggunakan pembobotan tf-idf dan cosine similarity"* penerapan *algoritme cosine similarity* digunakan untuk mengelompokan setiap berita agar sesui dengan kategori yang tersedia (Herwijayanti, 2017). Kemiripan antara *string* dihitung didapat dari duah buah *vector* yang saling dibandingkan. Dari hasil perhitungan tersebut akan menghasilkan suatu nilai kemiripan dari nol sampai satu, apabila nilai dari hasil tersebut nol maka data string tersebut dikatakan mirip, jika hasilnya satu maka data tersebut dikatakan tidak mirip (Ariantini, 2016). Dari hasil evalusi yang dihitung menggunakan rumus *cosine similarity* maka hasil tersebut dapat di gunakan untuk melakukan perangkingan, dimana *data* grup *facebook* yang memiliki *similarity* paling tinggi akan berada paling utama atau yang paling di rekomendasikan, kemudian diikuti nilai similarity yang ada di bawah sampai nilai *similarity* yang paling sedikit berada paling akhir. Dalam kasus ini *data string* yang dimaksut adalah *data string* dari grup *facebook* meliputi deskripsi, judul, *post* terakhir dan *data string* dari produk yang ada di *marketplace* meliputi judul produk, deskripsi dan kategori dari produk tersebut.

Dari permasalahan di atas dibutuhkanlah sebuah *sistem* yang dapat digunakan untuk melakukan jual beli secara *online*, dan juga sistem tersebut dapat difungsikan sebagai media promosi ke sosial media yang lebih mudah dan efisien. Penjual manambahkan data *grup* yang diinginkan ke sistem *marketplace* kemudian sistem tersebut dapat merekomendasikan grup *facebook* mana yang sesuai untuk mempromosikan produk tersebut. Dari penjelasan di atas, penelitian ini mengangkat judul tentang "Pembangunan *Sistem Marketplace* Yang Dapat Merekomendasikan Grup *Facebook* Yang Sesuai Dengan Produk Menggunakan Algoritme *Cosine Similarity* ". Penjual dipermudah dengan adanya algoritme *Cosine Similarity* yang akan langsung melakukan pencocokan antara produk dengan grup *facebook* yang memiliki tema yang sama sehingga hal tersebut dapat diharapkan munculnya calon-calon pembeli yang berpotensial.

1.2 Rumusan masalah

Bedasarkan latar belakang yang telah di jabarkan, maka rumusan masalah dapat di disimpulkan seperti berikut.

1. Bagaimana analisis kebutuhan pengembangan sistem *marketplace* dengan menerapkan *Graph API* menggunakan *Algoritme Cosine Similarity* ?
2. Bagaimana merancang sistem *marketplace* yang bisa terintegrasi dengan sosial media *facebook*?

3. Bagaimana metode yang dilakukan untuk menguji kinerja dari *sistem marketplace* yang bisa terintegrasi dengan *sosial media facebook*?

1.3 Tujuan

1. Analisis kebutuhan dilakukan menggunakan elisitasi kebutuhan dengan teknik observasi dan wawancara.
2. Mengembangkan layanan *Graph API Facebok* untuk mengintegrasikan sosial media *Facebook* dengan Sistem *Marketplace.*
3. Menguji kinerja dari sistem dengan menggunakan pengujian *white box* dan *black box*

1.4 Manfaat

Diharapkan penelitian ini akan memberikan manfaat sebagai berikut:

1. Bagi Penjual, aplikasi ini dapat digunakan sebagai saranan promosi atau penyebaran informasi dengan mudah. Dengan cara penjual memasukan data ke website kemudian secara otomatis barang dari penjual akan masuk ke website dan apabila data sudah lengkap penjual tinggal melakukan pembagian konten dari *website* ke sosial media sesuai rekomendasi dari hasil pemrosesan di sistem
2. Bagi peneliti, aplikasi atau *website* ini dapat dijadikan sebagai bahan pembelajaran dalam mengatasi masalah yang ada di masyarakat sekitar.

1.5 Batasan masalah

Dari rumusan masalah yang telah disampaikan di atas maka penelitian ini difokuskan dengan batasan masalah sebagai berikut:

1. Sistem yang dibangun nantinya berbasis *web*.
2. Pembangunan web menggunakan *framework codeigniter*.
3. *Graph API* digunakan untuk membantu penjual dalam membagikan informasi ke Sosial media *Facebook*.
4. Pembangunan sistem akan menggunakan *algoritme Cosine Similarity* yang berfungsi untuk melakukan rekomendasi grup yang sesuai dengan barang yang dijual.

.

1.6 Sistematika pembahasan

Untuk mencapai tujuan yang diinginkan dan memudahkan pembaca dalam memahami sistematika penulisan di dalam skripsi ini. Sistematika pembahasan penelitian ini dapat di uraikan sebagai berikut:

BAB I PENDAHULUAN

Bab pendahuluan memuat tentang latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika pembahasan dari PEMBANGUNAN SISTEM *MARKETPLACE* YANG DAPAT MEREKOMENDASIKAN *GRUP FACEBOOK* YANG SESUAI DENGAN PRODUK MENGGUNAKAN *ALGORITME COSINE SIMILARITY*.

BAB II LANDASAN KEPUSTAKAAN

Bab ini berisi mengenai kajian pustaka yang berhubungan atau terkait dengan penelitian yang telah ada. Menjelaskan teori-teori pendukung yang menjadi konsep dasaar dari penelitian ini.

BAB III METODOLOGI

Bab Metodologi berisi tentang metode, teknik, maupun langkah-langkah yang digunakan dalam studi literatur dan penlitian.

BAB IV ANALISIS DAN PERANCANGAN

Bab analisis dan perancangan ini berisi tentang analisis kebutuhan dan perancangan sistem yang diperlukan untuk membangun sistem *marketplace* dengan menerapkan *graph API algoritme Cosine Similarity*

BAB V IMPLEMENTASI

Bab implementasi ini berisi tentang implementasi sistem susai dengan perancangan yang sudah dilakukan di bab IV .

BAB VI PENGUJIAN

Bab ini memuat tentang pengujian dari sistem yang telah dibuat.

BAB VII PENUTUP

Pada Bab ini berisi tentang kesimpulan yang didapat dari hasil pengujian sistem yang telah dibuat serta saran untuk pengembangan sistem tersebut kedepannya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Pada penelitian ini penulis merujuk ke beberapa acuan yang relavan dan pernah dilakukan oleh Mansur pada tahun 2015 dan Angga kurnia Putra dan kawan-kawan pada tahuan 2017.

Pada penelitian Mansur (2015) bertujuan untuk membuat sebuah *marketplace* yang ditunjukan untuk promosi produk usaha kecil dan menengah. Pada penelitian yang dilakukan Mansur menggunakan konsep B2B. B2B merupakan konsep komunikasi bisnis secara elektronik antar perusahaan yang dilakukan secara rutin dan dalam kapasitas produk yang banyak atau besar besar.

Sedangkan pada penelitian yang dilakukan Angga kurnia dan kawan kawan(2017) yaitu penelitian membangun sebuah *marketplace* yang bertujuan untuk penyedia jasa les *private* berbasis *web*. Dimana sistem yang dibangun hanya ditunjukan bagi penyedia les *private* saja dan tidak diperuntukan untuk penjual produk dalam bentuk barang. Sehingga penelitian tersebut bertujuan untuk menyediakan *marketplace* yang hanya untuk penyedia jasa les *private* saja.

2.2 Waterfall Software Development

Metode *warterfall* pertama kali dikenalkan oleh Royce pada tahun 1970 dengan 7 (tujuh) tahapan yang berurut meskipun mempunyai *feedback loop* bila di perlukan.Menurut Pressman dalam bukunya *"Software Engineering"* metode waterfall atau yang sering disebut dengan *classic lifecyle* merupakan metode pengembangan perangkat lunak terstruktur yang paling banyak digunakan dan dikenal secara luas. Meskipun pada metode waterfall ini mengalami banyak perubahan, seperti perubahan langkah dari 7 (tujuh) menjadi 5 (lima).

Metode *Waterfall* paling terbaru adalah metode versi Sommervile (2011). Pada versi ini metode *Waterfall* sudah menggunakan 5 tahapan, tahapan tersebut meliputi:

1. *Requirements analysis and definition*

   Pada tahap ini adalah tahap konsultasi dengan pengguna dimana hasil dari konsultasi akan perjelas dan di definisikan secara rinci yang kemudian akan berfungsi sebagai spesifikasi sistem.

2. *System and software design*

   Pada tahap ini adalah tahap perencangan perangkat lunak yang melibatkan identifikasi dan penggambaran dari sistem dasar perangkat lunak dan hubunganya. Dalam tahap ini juga di gunakan untuk mentukan kebutuhan-kebutuhan sistem baik perangkat lunak maupun perangkat keras.

3. *Implementation and unit testing*

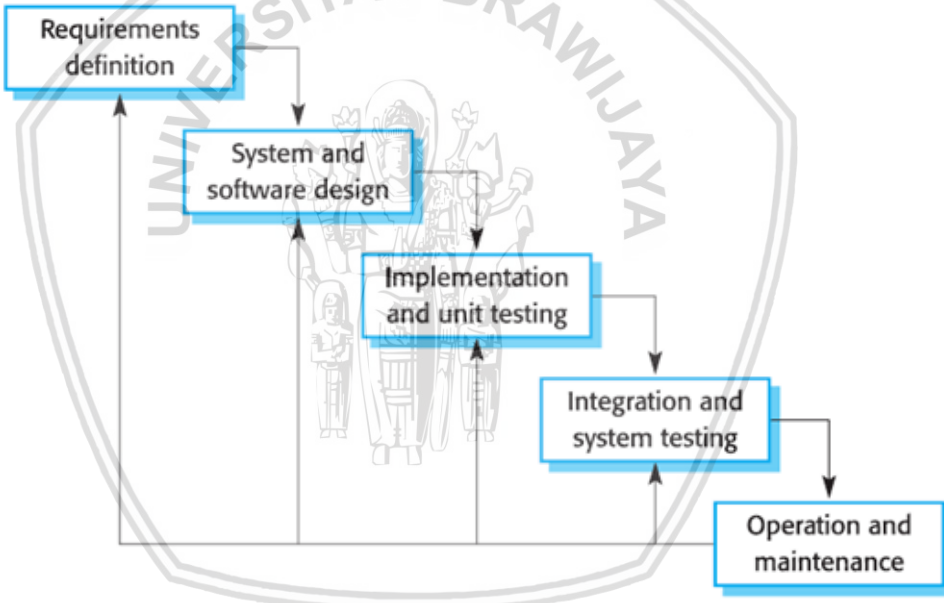
Tahapan ini merupakan tahapan untuk merealisasikan perancangan perangkat lunak unit. Dan juga pada tahap ini digunakan untuk melakukan verifikasi pada setiap unitnya agar sesuai dengan spesifikasinya.

4. *Integration and system testing*

   Pada tahapan ini adalah penggabungan pada unit-unit individu, unit tersebut di gabung hingga menjadi bentukk sistem lengkap. Dan juga setelah membentuk sebuah sistem yang lengkap kemudian dilakukan testing untuk memastikan apakah sudah sesuai dengan kebutuhan perangkat lunak.

5. *Operation and maintenance*

   Pada tahapan ini merupakan tahpan yang panjang, dimana setelah sistem di pasang dan digunakan oleh pengguna secara nyata, sistem akan di *Maintenance* untuk meningkatkan layanan sistem.



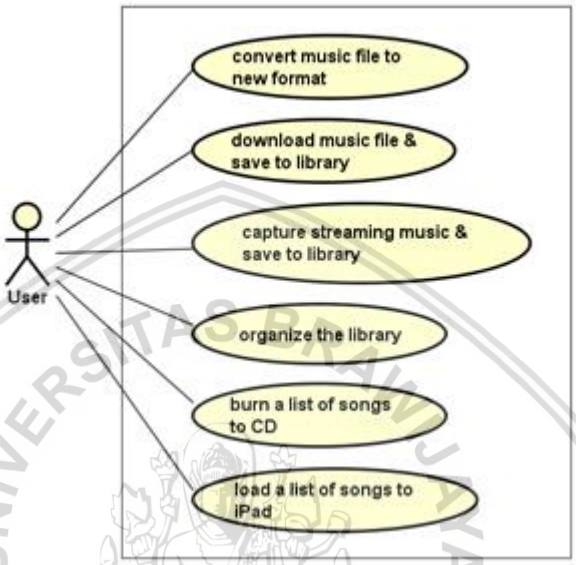
Gambar 2.1 Metode *Waterfall (Sommervile, 2011:30)*

2.3 *UML*

Unified Modeling Language atau yang disingkat dengan UML merupakan bahasa pemodelan perangkat lunak yang dirancang untuk memudahkan dalam memvisualisasikan rancangan dari sebuah sistem. Dibuat dan dikembangkan oleh Ivar Jacobson, Grady Booch, James Rumbaugh pada tahun 1994-1995 di *Rational Software* dengan dilanjutkan pengembangannya pada tahun 1996 (Haviludin, 2011).

2.3.1 Use Case

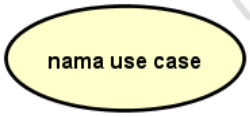
Use case diagram merupakan salah satu bahasa pemodelan UML. Diagram ini dapat membantu pengembang dalam menentukan fitur dan fungsi perangkat lunak dari sudut pandang pengguna. Diagram ini menggambarkan bagaimana pengguna berinteraksi dengan sistem dengan menggunakan langkah-langkah yang disediakan untuk memenuhi suatu tujuan (Pressman, 2010:847). Gambar 2.2 merupakan contoh dari Use case diagram.



Gambar 2.2 Contoh *Use Case Diagram (Pressman, 2010:847)*

Penjelasan Singkat tentang symbol-symbol pada *Use Case* akan ditunjukan pada tabel 2.1 Berikut:

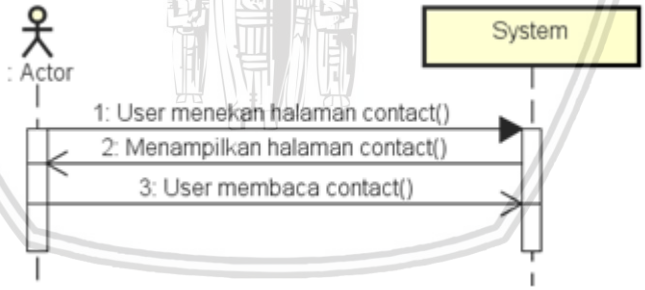
Tabel 2.1 Simbol *Use Case Diagram*

Simbol	Deskripsi
Use case 	Sebuah unit yang disediakan oleh sistem, dapat berhubungan dengan aktor atau unit-unit yang lain, dan memiliki fungsionalitas tersendiri. Pemberian nama *use case* umumnya memakai kata kerja.
Aktor 	Aktor adalah proses, orang, atau sistem lain yang berada diluar sebuah sistem. Aktor dapat berinteraksi dengan sistem. Pemberian nama aktor umumnya memakai kata benda pada awal frase.
Asosiasi _____	Mempresentasikan sebuah interaksi yang ada antara *use case* dan aktor.

Simbol	Deskripsi
Ekstensi <<extends>>	Mempresentasikan interaksi tambahan dari *use case* ke sebuah *use case* baru dimana suatu *use case* dapat berdiri sendiri tanpa adanya *use case* tambahan.
Generalisasi	Mempresentasikan interaksi umum-khusus antara dua *use case* dimana salah satu *use case* mempunyai fungsi lebih umum dari pada use case lainnya.
Include <<include>>	Mempresentasikan interaksi tambahan dari *use case* ke sebuah *use case* baru dimana suatu *use case* tidak dapat berdiri sendiri tanpa adanya *use case* tambahan. *Use case* tambahan mempunyai fungsi sebagai syarat dijalankan suatu *use case*.

2.3.2 Sequence Diagram

Sequence Diagram merupakan salah satu bahasa pemodelan UML. Diagram ini digunakan untuk menggambarkan tingkah laku objek pada *use case* dengan mendefinisikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Dalam penggambaran *sequence diagram* diperlukan objek-objek yang terlibat dalam sebuah *use case* dengan metode-metode yang diinstansiasi menjadi objek tersebut dalam suatu kelas. Banyaknya *sequence diagram* yang dibuat adalah minimal sebanyak pendefinisian *use case* yang telah didefinisikan. Contoh *sequence diagram* disajikan pada gambar 2.3 berikut ini.

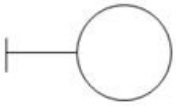
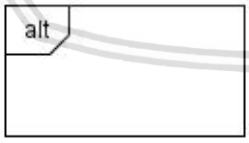


Gambar 2.3 Contoh *Sequence Diagram*

Penjelasan Simbol-simbol yang terdapat pada *Sequnce Diagram* ditunjukan pada tabel 2.3 berikut.

Tabel 2.2 Penjelasan simbol *Sequence Diagram*

Nama	Gambar	Deskripsi
Aktor		Aktor adalah proses, orang, atau sistem lain yang berada diluar sebuah sistem, Aktor juga dapat berhubungan dengan sistem. Pemberian nama aktor

Nama	Gambar	Deskripsi
		umumnya memakai kata benda pada awal frase.
Entity Class		Mempresentasikan informasi yang harus disimpan oleh sistem (struktur data dari sebuah sistem).
Boundary Class		Mempresentasikan hubungan antara satu atau banyak aktor dengan sistem, memodelkan bagian sistem yang bergantung pada pihak lain disekitarnya dan sebagai pembatas sistem dengan dunia luar.
Control		Mempresentasikan "perilaku mengatur", mengkoordinasikan perilaku sistem dan gerak dari suatu sistem, mengerjakan tugas utama dan mengontrol alur kerja suatu sistem.
Message		Mempresentasikan sepesifikasi dari hubungan antar objek yang menampung informasi tentang operasi yang terjadi.
Life Line		Mempresentasikan objek *entity*, antarmuka yang saling berhubungan.
Fragment		Suatu *fragment* mempresentasikan suatu potongan atau potongan hubungan (yang disebut operan interaksi) yang dikendalikan oleh suatu operator interaksi, yang bersesuaian kondisi-kondisi Boolean yang kenal sebagai batasan interaksi. Suatu fragment nampak dengan nyata sebagai jendela transparan, yang dibagi oleh bentuk garis horizontal untuk operan masing-masing.

2.4 Pengujian Perangkat Lunak

Kegiatan pengujian perangkat lunak adalah suatu aktivitas pemerikasaan yang dilakukan bertujuan untuk mendapatkan informasi mengenai kualitas perangkat lunak yang sedang diuji. Pengujian perangkat lunak merupakan proses menjalankan, meninjau, serta mengevaluasi sebuah perangkat lunak apakah telah memenuhi persyaratan atau belum. Pada penelitian ini kegiatan pengujian dilakukan dengan metode fungsional berupa pengujian *whitebox,* pengujian *blackbox*, dan metode non-fungsional berupa pengujian *compatibility* dan *security*.

2.4.1 Pengujian Fungsional

Pengujian fungsional merupakan pengujian yang berguna untuk pengujian perilaku yaitu menentukan apakah program melakukan apa yang seharusnya dilakukan berdasarkan kebutuhan fungsional. Kegiatan pengujian ini dilakukan dengan cara memberi masukan pada sistem dengan masukan nilai yang normal maupun tidak normal agar dapat menemukan berbagai kemungkinan kesalahan seperti *bug* yang akan muncul pada sistem (Shi, 2010).

2.4.1.1 Pengujian *White Box*

Pada pengujian *whitebox* adalah pengujian yang didasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur kontrol dari *desain* program secara *procedural* untuk membagi pengujian ke dalam beberapa kasus pengujian. Pada pengujian *whitebox* digunakan untuk pengujian unit dan pengujian integrasi. Pengujian unit merupakan pengujian yang berguna untuk memeriksa modul, modul yaitu inti terkecil dari perangkat lunak. Dengan menggunakan gambaran dari rancangan mekanisme sebagai petunjuk, jalur kontrol yang penting diuji untuk mendeteksi kesalahan dalam batas modul tersebut. Pengujian unit mengarah pada *white-box* dan langkahnya dapat dilakukan secara paralel untuk model bertingkat (Pressman, 2010:485). Pengujian integrasi lebih pada pengujian penggabungan dari dua atau lebih unit pada perangkat lunak.

2.4.1.2 Pengujian *Black Box*

Pengujian *blackbox* merupakan pengujian yang berguna untuk menguji fungsi-fungsi khusus dari perangkat lunak yang dirancang. Pada pengujian ini penulis menggunakan metode *Equivalence Partitioning*. Metode *Equivalence Partitioning* adalah metode pada pengujian *black-box* yang melakukan pembagian domain masukan atau inputan dari suatu program kedalam bentuk kelas-kelas data, dimana kasus uji dapat diturunkan (Pressman, 2010). Menurut Pressman dalam bukunya "*Software Engineering*". Klas *Equivalence* dapat didefinisikan sesuai dengan panduan berikut:

1. Jika inputan atau masukan mempunyai jenjang tertentu , maka kondisi *valid* dan tidak *valid* harus didefinisikan.

2. Jika inputan atau masukan membutuhkan nilai tertentu, maka kondisi valid dan tidak valid harus didefinisikan.
3. Jika inputan atau masukan membutuhan grup atau himpunan tertentu, maka kondisi valid dan tidak valid harus didefinisikan.
4. Jika inputan atau masukan kodisinya boolean, maka kondisi *valid* dan tidak *valid* harus didefinisikan.

Tabel 2.3 *Equivalence Partitioning*

Input nilai	Invalid	valid
Input String dari A – Z	-	a-z
Input nilai Numerik 0 – 9	-	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Input nilai numerik -1 s/d -9		-1, -2, -3, -4, -5, -6, -7, -8, -9
Input nilai char @, ?, ! ~	-	@, ?, ! ~
Input nilai null	-	Null

2.5 Algoritme Cosine Similarity

Algoritme *Cosine Similarity* adalah sebuah algoritme yang berfunsi untuk mengukur tingkat kemiripan dari dua objek. *Cosine Similarity* dapat menyamakan frekuensi kata pada kalimat menggunakan persamaa *Term Frequency (TF)*. *Term Frequency* akan memisah setiap kalimat menjadi kumpulan kata, hal tersebut bertujuan untuk mempermudah dalam menyamakan kedua kalimat atau kata yang nantinya akan di bandingkan tingkat kemiripannya.

Secara umum penentuan tingkat kemiripan satu kalimat atau dokumen dengan *query* di pandang sebagai pengukuran (*similarity measure*). Semakin tinggi tingkat kemiripan suatu kalimat atau dokumen maka hal tersebut akan semakin sesuai dengan *query*. Persamaan yang di gunakan untuk menghitung *cosine similarity* .

$$\frac{A \,.\, B}{\|A\| \, \|B\|} \quad \frac{\sum_{i-1}^{n} A_i x B_i}{\sqrt{\sum_{i-1}^{n} (A_i)^2} \; x \; \sqrt{\sum_{i-1}^{n} (B_i)^2}}$$

Dengan keterangan :

A = vektor

B = vektor

Ai = bobot *term* i dalam blok Ai

Bi = bobot *term* i dalam blok Bi

i = Jumah *term* dalam kalimat

n= jumah vektor

Dimana A merupakan bobot setiap ciri pada *vektor* A, dan B merupakan bobot setiap ciri pada B. jika dikaitkan dengan *information retrieval* maka A adalah bobot setiap istilah pada *grup facebook*, dan B merupakan bobot setiap istilah pada produk. Pada penelitian ini digunakan *cosine similarity* untuk mengukur kemiripan antar dokumen. Pengukuran kemiripan dapat dilakukan dengan membandingkan dokumen 1 dengan dokumen 2 kemudian sistem akan menghitung nilai kemiripan. Ai.Bi adalah nilai yang diperoleh dari *term* A dan *term* B kemudian kedua nilai tersebut dijumlahkan, kemudian nilai Ai2 semua nilai *term* dokumen A semuan nilainya dipangkatkan dua, begitu juga dengan *term* Bi2 semua nilai yang diperoleh dipangkatkan dua kemudian semua nilai yang diperoleh dijumlahkan.

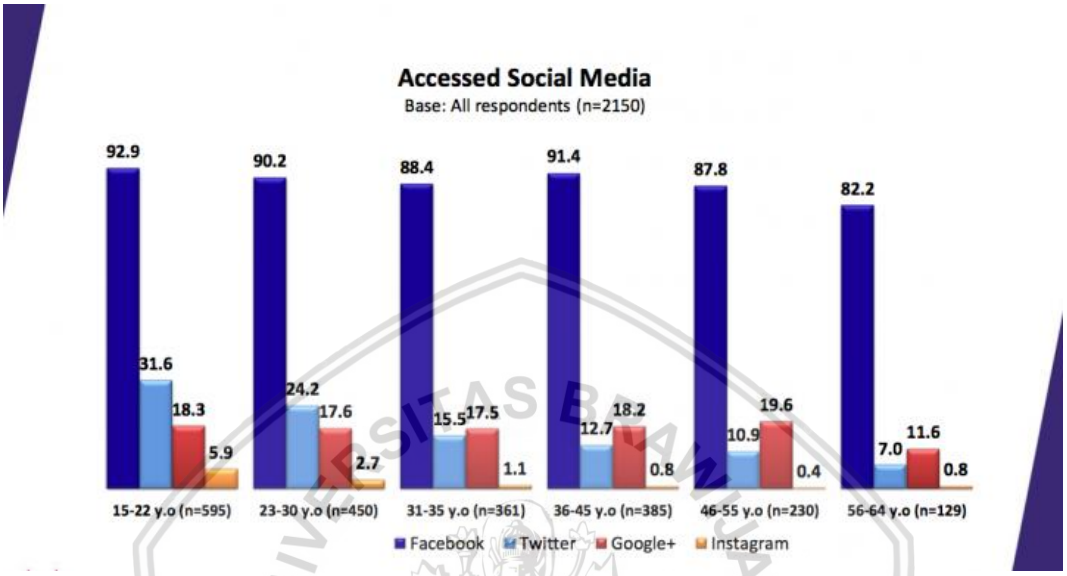
2.6 Marketplace

Marketplace merupakan pasar secara *virtual* berbasis *internet* dimana setiap produk yang ingin dijual dapat diperkenalkan dan melakukan transaksi barang atau jasa. *Marketplace* juga dapat dikategorikan sebagai hubungan transaksi secara langsung maupun tidak langsung sehingga *marketplace* dapat mempermudah penjual produk untuk menyampaikan *detail* dari informasi dari produk yang di jual atau mereka pasarakan. *Marketplace* meurupakan sebuah tool dalam melakukan perantara informasi antara organisasi yang memungkinkan pembeli dan penjual dapat berpartisipasi dalam melakukan jual beli(Zheng, et al, 206).

2.7 Facebook Graph API

Facebook diciptakan pada tahun 2003, penciptanya yaitu Mark Zuckerberg yang merupakan salah satu mahasiswa *Harvard University* pada saat itu. *Facebook* yang awalnya muncul dengan alamat TheFacebook.com dan hanya memiliki pengguna dari mahasiswa kampus itu sendiri. Seiring berjalanya waktu *facebook* mengalami peningkatan pada tahun 2010, bahkan pada saat itu *facebook* mengalahkan jumlah pengunjung *Google*. *Facebook* juga menjadi jejaring sosial teratas dikawasan Asia yaitu Malaysia, Australia, Filipina, Singapura, Hong Kong, Vietnam, Selandia Baru dan juga Indonesia (Ekawati, 2012).

Saat ini *Facebook* merupakan sosial media yang paling populer untuk saat ini. Seperti *survei* yang dilakukan oleh Enricko Lukman pada tahun 2013, data tersebut menunjukkan bahwa *facebook* menjadi sosial media yang paling dominan dari pada lainya seperti , *Google plus*, *Twitter*, dan *Instagram* (Lukman. 2013). Oleh karena itu *facebook* sampai saat ini masih sangatlah berpengaruh apabila digunakan untuk promosi barang atau jasa. Seperti gambar 2.4 berikut:

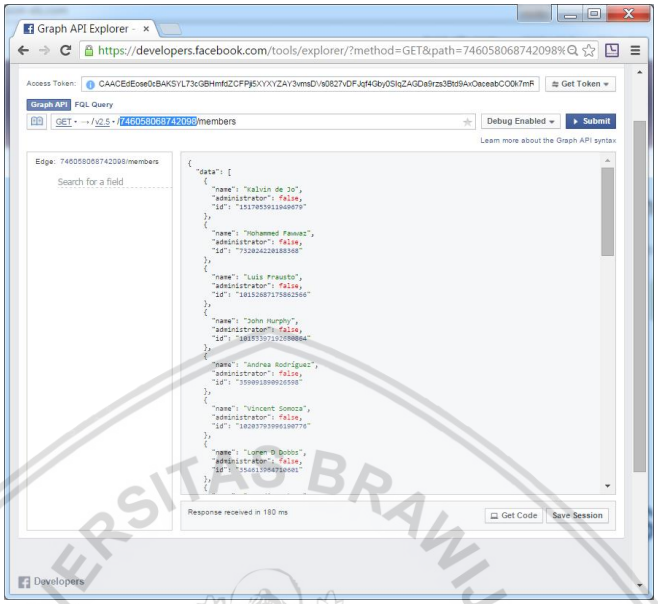


Gambar 2.4 *Dominasi Facebook* tahun 2013 *(Lukman, 2013)*

Dengan berkembangnya *facebook*, maka *facebook* memberikan fasilitas *Facebook Graph API* bagi para pengembang yang ingin menggunakan data di *Facebook*. *Facebook Graph API* adalah sebuah cara pemanggilan atau pengiriman data dari *facebook* mengenai *user* dan dapat juga digunakan untuk memanggil dan mengirim data dalam bentuk proses menggunakan *social plugin*. Data user dapat dipanggil hanyalah data yang sudah mendapatkan ijin hak akses dari pemilik akun *facebook* untuk pertama kalinya menggunakan aplikasi *API* tersebut. Dengan menggunakan *Graph API query* pemanggilan akan lebih disederhanakan dalam bentuk path sehingga *developer* akan lebih mudah dalam pemanggilan data yang kompleks.

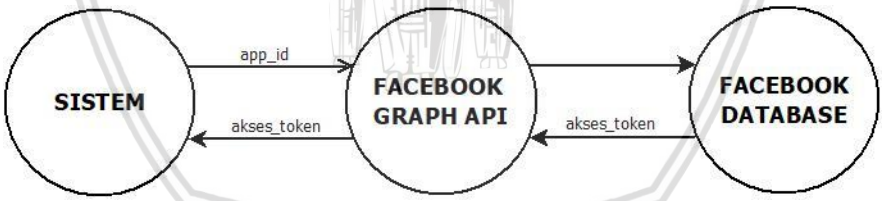
Gambar 2.5 Alur dari *Facebook graph* (Kaur,2014)

Dari hasil *facebook Graph API output* yang dihasilkan adalah dalam bentuk *Java Script Object Natation (JSON)*. Dari hasil *ouput Facebook Graph API* tersebut kemudian diproses dalam beberapa metode, yaitu metode menghapus, minta data dan memposting aktifitas(kaur, 2014).



Gambar 2.6 *Output Graph API* (kaur, 2014)

Pada untuk penerapanya *Facebook Graph API* akan disematkan pada sistem, sehingga untuk alunya, sisitem akan mengirimkan *app_id* yang telah kita buat dengan menggunakan *Graph API* tadi kemudian akan diteruskan ke *Facebook Database,* setelah itu *Facebook Database* akan mengirimkan lagi akses token yang diminta.



Gambar 2.7 *Alur Graph API*

Pada penerapan ini mengunakan graph api versi 2.9 dan 2.12

2.8 Codeigniter

Codeigniter adalah aplikasi *open source* berupa *framework* dengan *model* MVC (*Model*, *View*, *Controller*) untuk membangun sebuah aplikasi berbasis web dengan menggunakan bahasa pemrograman *PHP*. Codeigniter dapat membantu para pengembang aplikasi untuk membuat aplikasi *web* dengan cepat tanpa harus membuatnya dari awal. *Codeigniter* sendiri dirilis pertama kali pada 28 februari 2006.

2.9 Boostrap

Bootstrap adalah sebuah *framework* yang dapat menyelesaikan permasalahan dalam mendesain web. Slogan dari *framework* boostrap adalah *"Sleek, intuitive, and powerful front-end framework for faster and easier web development",* yang artinya kita dapat mendesain sebuah website dengan lebih rapi, cepat dan mudah (Ahmad.2015). Boostrap sendiri merupakan sebuah *platfom* yang *responsive* artinya boostrap ini bila di terapkan pada sebuah *website* dan *website* tersebut dibuka dengan *smartphone* ataupun desktop maka tampilannya tetap rapi dan baik.

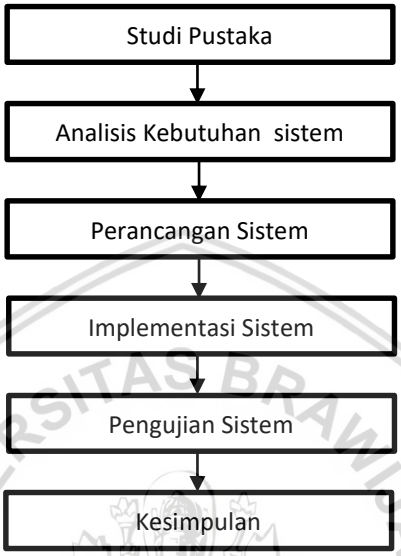
Framework boostrap sudah meluas dikalangan *front-end* web. Perkembangannya pun sampai sekarang masih terus berlangsung. Untuk penggunaan framework ini tidaklah terlalu rumit. Hanya dengan memanggil *Cascading Style Sheet (CSS)* dan *Java Script (JS)* yang teradapat dalam boostrap lalu menuliskan kelasnya di dalam kodingan yang ingin diterapkan boostrap.

Pada *framework* boostrap juga terdapat *grid system*. *Grid system* merupakan pengaturan ukuran yang di tampikan pada tampilan nantinya. *Grid system* sendiri berfungsi untuk mengatur lebar dari tampilan yang ingin diterapkan boostrap. Dari *grid system* ini kita bebas untuk melakukan pengaturan *responsive* yang kita inginkan. Boostrap memiliki 12 grid yang dapat kita gunakan untuk mengatur responsive web yang kita bangun, setiap grid dari 12 grid tersebut memiliki fungsi masing-masing. Berikut gambaran *grid system* pada *boostrap* pada gambar 2.8

Gambar 2.8 *Grid system Boostrap* (**Sumber**: coderomeos.org)

BAB 3 METODOLOGI

Pada bab ini menjelaskan tentang langkah-langkah yang akan di lakukan dalam implementasi aplikasi yang akan di buat bedasarkan data-data yang telah di kumpulkan . berikut ini diagram urutan dalam pengerjana perangkat lunak yang telah di rencanakan sesuai dengan data yang ada.



Gambar 3.1 Metodologi Penelitian

3.1 Studi Pustaka

Pada bagian ini menjelaskan tentang metode-metode yang di gunakan untuk mendapatkan teori-teori untuk mengerjakan pembuatan aplikasi ini dan untuk penulisan skripsi. Pustaka yang berhubungan dengan penulisan skripsi ini diantaranya:

1. *Sistem Development Life Cycle (SDLC)*
2. *Unified Modeling Language (UML)*
3. Pengujian Perangkat Lunak
4. *Algoritme Cosine Similarity*
5. *Marketplace*
6. *Facebook Graph API*
7. *Codeigniter*

3.2 Analisis Kebutuhan

Analisis kebutuhan digunakan untuk mendapatkan semua kebutuhan yang akan digunakan dalam pengembangan sistem *marketplace* yang dapat merekomendasikan *grup facebook* yang sesuai dengan produk menggunakan

algoritme cosine similarity. Dimana analisis kebutuhan ini mengacu pada hasil dari elisitasi dan juga kebutuhan sistem. Kebutuhan yang telah dihasilkan akan dimodelkan dalam identifikasi aktor dan kebutuhan fungsional beserta kebutuhan *non* fungsional. Dari hasil indentifikasi aktor dan kebutuhan fungsional serta non fungsional akan dimodelkan dalam bentuk *use case diagram* dan akan dijelaskan dalam bentuk *Skenario Use case*

3.3 Perancangan Sistem

Setelah melakukan tahapan analisis kebutuhan kemudian di lanjutkan dengan tahapan perancangan. Perancangan merupakan tahapan dalam pembangunan sistem. Tahapan peracangan ini akan menggunakan *sequence diagram, class diagram* untuk pemodelan perancangan kebutuhan, perancangan algoritme untuk merancang kode yang akan diimplementasikan dan perancangan antarmuka digunakan untuk merancang tampilan atau antarmuka pada sistem yang akan dibangun.

3.4 Implementasi Sistem

Setelah melakukan tahap perancangan maka selanjutnya yaitu tahap implementasi. Tahap implementasi ini merupakan tahapan yang mewujudkan suatu konsep hasil analisis kebutuhan dan perancangan dari suatu bentuk dokumen ke dalam bentuk aplikasi yang siap digunakan. Tahap implementasi meliputi:

a Implementasi sistem *marketplace* menggunakan *pemograman* PHP dan juga *javascript*, untuk bagian antarmuka sistem menggunakan *framework* Boostrap, sedangkan untuk membantu pengembangan sistem menggunakan *framework codeigniter* yang berfungsi untuk mempermudah dalam pengembangan sistem.

b Untuk implementasi basis data menggunakan DBMS Mysql pada server localhost (WAMPP).

3.5 Pengujian sistem

Pada tahap selanjutnya yaitu tahap pengujian dan analisis . tahap ini berfungsi untu parameter pada sistem yang sudah dibuat sudah sesuai dengan analisis kebutuhan atau tidak. Dalam penelitian ini pengujian yang digunakan adalah pengujian whitebox dan blackbox. Pengujian whitebox digunakan untuk pengujian unit dan pengujian integrasi. Pengujian blackbox digunakan untuk pengujian *equivalence partitioning*.

1. Pengujian unit merupakan pengujian dengan menguji unit terkecil dari sistem untuk mengetahui apakah unit sistem tersebut dapat digunakan atau tidak. Dalam penelitian ini pengujian unit dilakukan menggunakan teknik *basis path testing. Basis path testing* digunakan untuk menguji kode program berdasarkan algoritme pada setiap metode yang ada di klas.

2. Pengujian integrasi digunakan untuk menguji dua unit atau lebih yang saling berhubungan. Metode yang digunakan yaitu pengujian *whitebox*. Teknik yang digunakan dalam penelitian ini adalah *basis path testing*. *Basis path testing* untuk menguji kode program berdasarkan algoritme pada setiap metode yang ada di klas.

3. Pengujian equivalence partitioning adalah metode pada pengujian black-box yang melakukan pembagian domain masukan atau inputan dari suatu program kedalam bentuk kelas-kelas data, dimana kasus uji dapat diturunkan (Pressman, 2010).

Dari hasil pengujian tersebut akan dilakukan analisis untuk mengetahui apakah sistem telah berjalan sesuai kebutuhan yang diinginkan. Jika belum memenuhi maka akan dilakukan perbaikan, mulai dari tahap analisis sampai pada implementasi dan pengujian.

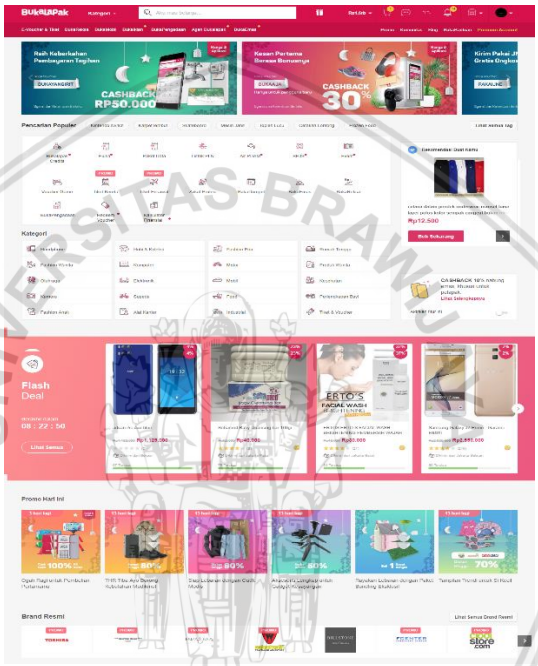
3.6 Kesimpulan

Kesimpulan dilakukan setelah semua tahap perancangan, implementasi dan pengujian telah selesai dilakukan. Penarikan kesimpulan diambil dari hasil pengujian dan analisis metode yang telah terapkan. Pada tahapa akhir adalah saran yang berujuan untuk memperbaikai kesalahan-kesalahan yang terjadi serta memberikan memberikan pertimbangan atas metode selanjutnya.

BAB 4 ANALISIS KEBUTUHAN DAN PERANCANGAN

4.1 Elisitasi kebutuhan

Elsitasi adalah tahapan penggalian kebutuhan yang dugunakan untuk menentukan kebutuhan apa saja yang harus ada pada sistem yang dikembangkan. Pada tahap ini bertujuan untuk membantu pengembangan dalam memahami sistem yang akan dispesifisikan. Pada proses elisitasi kebutuhan ini menggunakan teknik observasi. Observasi dilakukan pada sistem *marketplace* yang sudah ada yaitu sistem dari *marketplace* bukalapak. Berikut gambar 4.1 dari sistem *marketplace* bukalapak



Gambar 4.1 Halaman utama pada sistem bukalapak (sumber: www.bukalapak.com)

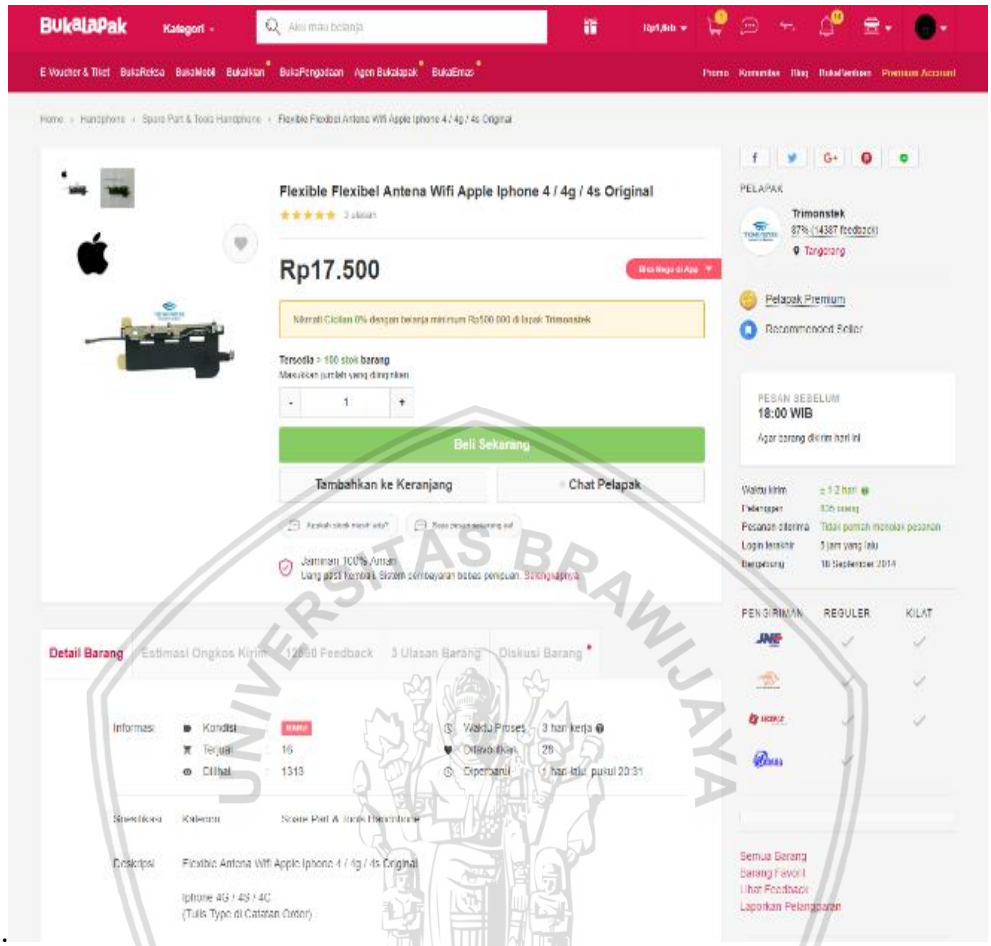
Pada gambar 4.1 menunjukan bahwa pada halaman tersebut terdapat beberapa bagian fungsi, mulai bagian navbar terdapat menu kategori, form pencarian, dan beberapa *icon* transaksi.

Pada gambar 4.2 menunjukan tampilan detail produk, pada bagian tampilan detail produk ini menampilkan gamabar produk, harga produk, ulasan produk, informasi produk dan lain-lain.

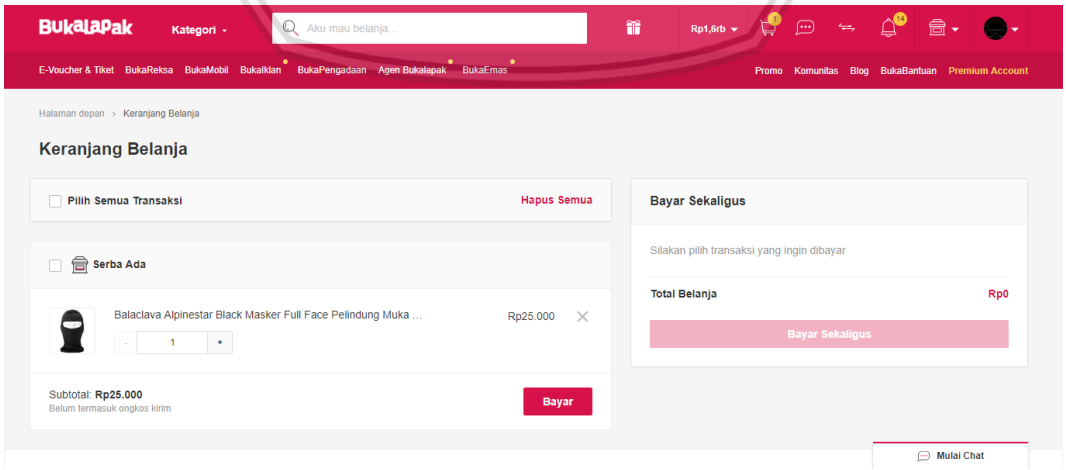
Pada gambar 4.3 merupakan gambar keranjang belanja pada sisi pembeli. Pada keranjang belanja ini berfungsi untuk menampilkan daftar produk yang sudah dimasukkan pada keranjang belanja untuk dilakukan pembelian produk. Pada keranjang belanja ini terdapat gambar produk, nama produk dan harga produk yang telah ditambahkan.

Pada gambar 4.4 merupakan halaman transaksi dari sisi pembeli. Halaman ini berfungsi untuk menampilkan barang yang sedang kita beli. Pada halaman ini

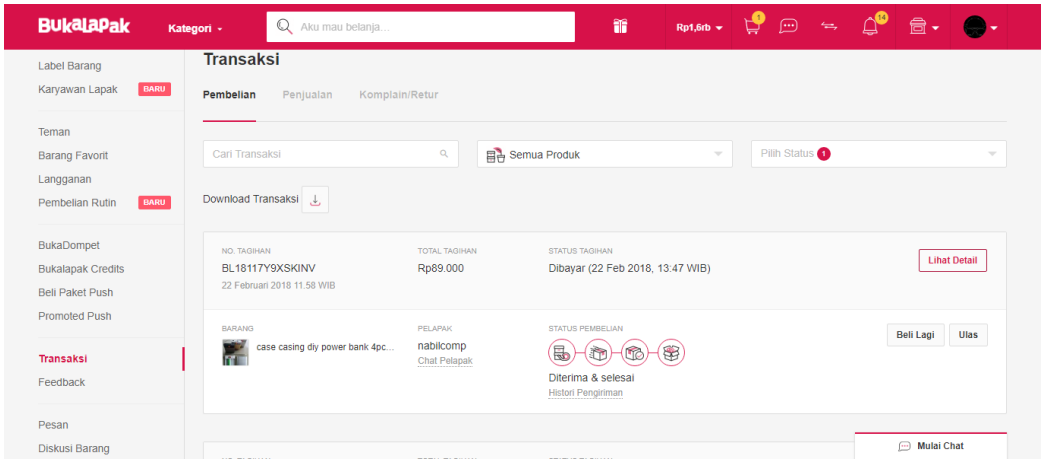
terdapat, nama barang, gambar barang, no tagihan, total tagihan, nama penjual, status pembayaran dan status pembelian.



Gambar 4.2 Halaman Detail produk sistem bukalapak (sumber: www.bukalapak.com)

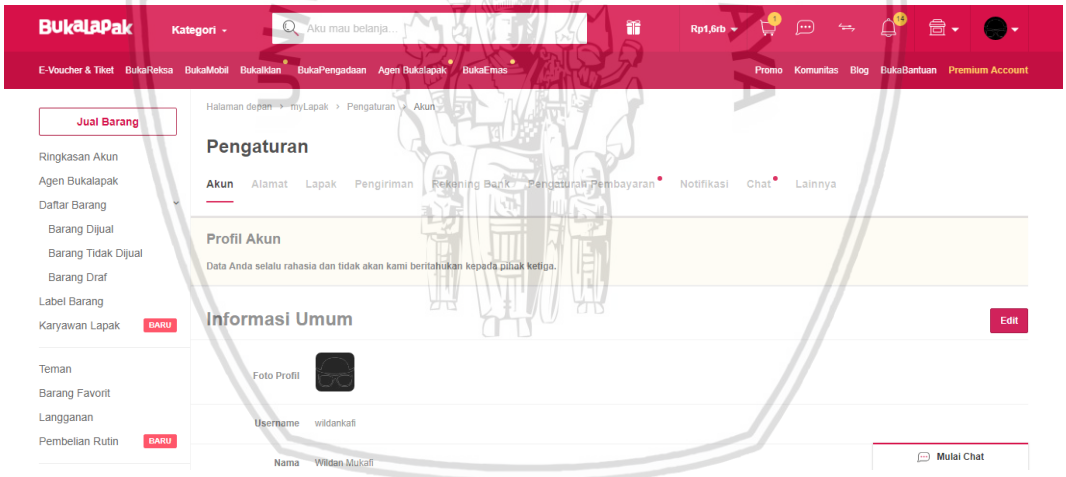


Gambar 4.3 Halaman Keranjang belanja (sumber: www.bukalapak.com)



Gambar 4.4 Halaman transaksi dari sisi pembeli (sumber: www.bukalapak.com).

Pada gambar 4.5 merupakan halaman pengaturan akun. Pada halaman ini berfungsi untuk merubah data dari akun yang kita punya, yang dapat dirubah dari pengaturan ini meliputi gambar profil, nama, username, email, alamat, rekening, kurir pengirim, email, password, catatan penjual dan lain-lainya. Pengaturan ini dapat dilakukan dari sisi penjual ataupun pembeli.



Gambar 4.5 Halaman pengaturan akun (sumber: www.bukalapak.com).

Pada gambar 4.6 merupakan fitur dari sisi penjual yang ingin menambah produk yang ingin ditambahkan, pada fitur tersebut terdapat beberapa formulir yang harus dilengkapai jika ingin melakukan tambah produk yang akan dijual pada sistem *marketplace.* Formulir tersebut meliputi nama produk, gambar produk, harga produk, kategori produk, berat produk dan lain sebagianya.

Gambar 4.6 Halaman tambah produk yang dijual (sumber: www.bukalapak.com)

Dari hasil observasi tersebut dapat diambil beberapa fitur yang harus terdapat pada sistem *marketplace* yang akan dibangun. Fitur-fitur tersebut meliputi:

Tabel 4.1 Fitur yang didapat dari hasil obeservasi

NO	Fitur sistem *marketplace*
1	Sistem harus menyediakan fungsi daftar
2	Sistem harus menyediakan fungsi masuk
3	Sistem harus menyediakan fungsi tampil data produk
4	Sistem harus menyediakan fungsi tambah produk baru
5	Sistem harus menyediakan fungsi keranjang belanja
6	Sistem harus menyediakan fungsi pengaturan akun
7	Sistem harus menyediakan fungsi transaksi

Bedasarkan observasi yang telah dilakukan maka penulis berinisiatif merancang suatu sistem *marketplace* yang terdapat fitur promosi ke *sosial media facebook* terutama pada bagian *grup facebook.* Dengan merancang sistem *marketplace* yang mengacu pada penelitian sejenis yang pernah dilakukan dan juga bedasarkan aktifitas sistem *marketplace* yang sudah ada. Maka sistem *marketplace* harus memiliki beberapa fungsi utama dari sistem *marketplace* yang sudah ada pada saat ini. Fungsi utama tersebut sesuai pada tabel 4.1 diatas. Kemudian terdapat fitur tambahan yang menerapkan *algoritme cosine similarity* untuk melihat kemiripan antara produk yang meliputi nama produk, kategori produk dan juga deskripsi dengan *grup facebook* yang meliputi nama grup facebook, deskripsi dan postingan pada *grup facebook*. Dan juga terdapat fitur post ke banyak *grup* secara langsung sesuai dengan nilai tingkat kemiripan yang dihasilkan dari *algoritme cosine similarity.*

4.2 Analisis kebutuhan

Pada bagian analisis kebutuhan ini akan melakukan identifikasi kebutuhan sistem yang akan dibuat. Identifikasi tersebut meliputi identifikasi aktor yang terlibat dalam sistem yang akan dibuat, dan juga mengidentifikasi kebutuhan fungsional dan non fungsional, membuat daftar kebutuhan dengan menganalisis kebutuhan dari aplikasi yang akan di buat.

4.2.1 Identifikasi aktor

Pada bagian tahap indentifikasi aktor ini berfungsi untuk melihat siapa saja yang terlibat pada sistem *marketplace* yang dapat merekomendasikan *grup facebook* yang sesuai dengan produk menggunakan algoritme *cosine similarity*. Pada tabel 4.1 berikut akan menjelaskan aktor beserta deskripsi yang terkait dengan aktor pada sistem.

Tabel 4.2 Identifikasi aktor

Aktor	Deskripsi
Admin (AD)	Admin merupakan pengguna yang akan mengelola data dan juga melakukan pengecekan pada setiap transaksi di sistem *marketplace*.
Pembeli (PB)	Pembeli merupakan pengguna yang dapat melakukan aktifitas pembelian barang yang telah dipasarkan oleh penjual pada sistem *marketplace*.
Penjual (PJ)	Penjual merupakan pengguna yang dapat melakukan aktifitas penjualan produk mereka mereka melalui sistem *marketplace*.
User (U)	*User* merupakan pengguna yang dapat melakukan beberapa aktifitas pada sistem *marketplace* tanpa melakukan login

4.2.2 Kebutuhan Fungsional Sistem

Pada bagian kebutuhan fungsional membahas mengenai layanan yang harus disediakan dalam sistem. Dan bagaimana reaksi sistem terhadap input atau aktifitas lain yang harus dilakukan sistem pada situasi tertentu. Langkah awal dapat dilakukan dengan mengidentifikasi apa saja kebutuhan yang harus dilakukan oleh sistem kemudian pada setiap kebutuhan akan *divalidasi* dan *diverifikasi*. Dari hasil *validasi* akan menunjukan apakah kebutuhan fungsional tersebut telah benar dan lengkap sesuai dengan kebutuhan pengguna. Sedangkan hasil dari *verifikasi* akan menunjukan bahwa kebutuhan fungsional dalam sistem yang dibangun dapat dengan mudah dipahami, identifikasi dan komplit serta tidak ada ambiguitas.

Setiap kode kebutuhan pada penelitian ini akan diberikan kode SM_F_XX. XX menunjukan nomor fungsi kebutuhan utama. Berikut daftar kebutuhan fungsional beserta spesifikasi kebutuhan yang harus ada pada sistem:

Tabel 4.3 Tabel kebutuhan Fungsional

Kode fungsi	Nama Fungsi	Deskripsi/Spesifikasi	Pengguna
SM_F_01	Masuk admin	Sistem memiliki fitur *login* untuk masuk ke bagian admin *panel* **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan halaman *login* atau masuk untuk masuk kehalaman dasboard admin yang	AD

Kode fungsi	Nama Fungsi	Deskripsi/Spesifikasi	Pengguna
		berisi form *username* dan *password*	
SM_F_02	Tampilkan data member	Sistem harus mampu menyediakan fungsi tampil daftar data member dalam bentuk tabel **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan daftar member yang berisi *username*, nama member *status*, tanggal bergabung dan tombol *action detail member* dalam bentuk tabel	AD
SM_F_03	Perbarui status member	Sistem harus mampu menyediakan fungsi memperbarui status member. **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menyediakan tombol aktif atau *non* aktif pada *detail* member yang berfungsi untuk memperbarui status *member* dari aktif jadi tidak aktif atau sebaliknya.	AD
SM_F_04	tampilkan daftar produk admin	Sistem harus mampu menyediakan fungsi tampil daftar produk **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan data produk yang berisi judul_produk, kategori produk, status produk dan tombol *action*	AD

Kode fungsi	Nama Fungsi	Deskripsi/Spesifikasi	Pengguna
		detail produk dalam bentuk tabel	
SM_F_05	Perbarui status produk	Sistem harus mampu menyediakan fungsi perbarui status produk. **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan status dari produk 2. Sistem harus mampu menyediakan tombol aktif atau non aktif pada halaman detail produk yang befungsi untuk memperbarui status produk dari aktif menjadi tidak aktif atau sebaliknya	AD
SM_F_06	tampilkan data produk kategori	Sistem harus mampu menyediakan fungsi tampil data kategori produk **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan halaman tampil data kategori produk yang berisi judul_kategori dan tombol *action edit* kategori ditampilkan dalam bentuk tabel	AD
SM_F_07	tambah data produk kategori	Sistem harus mampu menyediakan fungsi tambah data kategori produk **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menyediakan formulir judul_kategori, formulir *icon* kategori	AD

Kode fungsi	Nama Fungsi	Deskripsi/Spesifikasi	Pengguna
		dan tombol submit untuk menyimpan data kategori baru pada halaman tambah *data* kategori	
SM_F_08	Perbarui data produk kategori	Sistem harus mampu menyediakan fungsi perbarui data kategori produk. **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan tombol *edit data* kategori produk. 2. Sistem harus mampu menampilkan form nama kategori produk dan *icon* kategori produk dalam bentuk *form input*	AD
SM_F_09	Konfirmasi Transaksi Produk Admin	Sistem harus mampu menyediakan fungsi konfirmasi transaksi produk oleh Admin **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan detail dari transaksi produk yang berisi , nama produk, jumlah, dikirim dari, penjual produk, harga produk dan foto produk. 2. Sistem harus mampu menampilkan tombol konifirmasi transaksi diterima atau transaksi ditolak.	AD

Kode fungsi	Nama Fungsi	Deskripsi/Spesifikasi	Pengguna
SM_F_10	tampilkan data grup *facebook*	Sistem harus mampu menyediakan fungsi tampil data grup *facebook* **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan daftar halaman grup yang berisi nomor, tag, id_grup, nama grup dan tombol *action detail* grup dalam bentuk tabel	AD
SM_F_11	Tampil daftar data User admin	Sistem harus mampu menyediakan fungsil tampil daftar data user admin **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan halaman tampil data user admin yang berisi username nama, status dan tombol *action detail user admin* dalam bentuk tabel.	AD
SM_F_12	Tambah user admin	Sistem harus mampu menyediakan fungsi tambah user admin **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menyediakan *form* tombah *user* admin yang berisi form *username*, *form email* dan *form password*.	AD
SM_F_13	Perbarui data user admin	Sistem harus mampu menyediakan fungsi perbarui *data user* admin **Spesifikasi Kebutuhan:**	AD

Kode fungsi	Nama Fungsi	Deskripsi/Spesifikasi	Pengguna
		1. Sistem harus mampu menyediakan formulir *username*, nama, *email*, alamat dan *status* pada halaman perbarui *data user* admin	
SM_F_14	Daftar	Sistem harus mampu menyediakan fungsi daftar **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menyediakan formulir nama depan, nama belakang, email, jenis kelamin, *username* dan *password* pada halaman daftar	U
SM_F_15	Masuk	Sistem harus mampu menyediakan fungsi masuk ke dasboard sistem. **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menyediakan formulir *username* dan *password* pada halaman masuk	PB, PJ
SM_F_16	Lihat katalog produk	Sistem harus mampu menyediakan fungsi lihat katalog produk **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan halaman katalog produk yang berisi nama produk, harga produk, gambar produk, ulasan produk dan jumlah produk.	U, PB,PJ

Kode fungsi	Nama Fungsi	Deskripsi/Spesifikasi	Pengguna
SM_F_17	Lihat *detail* produk	Sistem harus mampu menyediakan fungsi lihat detail produk **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan halaman detail produk yang berisi judul produk, jumlah produk, berat produk, jenis produk, garansi, total terjual, terlihat, kurir, deskripsi, catatan penjual dan ulasan produk.	U, PB,PJ
SM_F_18	Lihat daftar produk kategori	Sistem harus mampu menyediakan fungsi lihat daftar kategori produk **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan daftar kategori yang berisi nama_kategori	U, PB, PJ
SM_F_019	Pencarian nama produk	Sistem harus mampu menyediakan fungsi pencarian nama produk. **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menyediakan formulir pencarian yang disikan dengan kata kunci nama produk yang akan dicari 2. Sistem harus mampu menampilkan hasil pencarian yang berisi nama produk, harga produk dan jumlah produk.	U, PB,PJ

Kode fungsi	Nama Fungsi	Deskripsi/Spesifikasi	Pengguna
SM_F_20	Tambah produk	Sistem harus mampu menyediakan fungsi tambah produk **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menyediakan formulir judul produk, kategori produk, sub kategori produk, super sub kategori produk, deskripsi produk, foto produk, jumlah produk, harga produk, berat produk, tipe produk, garansi produk dan foto utama produk pada halaman tambah data produk	PJ
SM_F_21	Perbarui produk	Sistem harus mampu menyediakan fungsi perbarui produk **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menyediakan formulir judul produk, kategori produk, sub kategori produk, super sub kategori produ, deskripsi produk, foto produk, jumlah produk, harga produk, berat produk, tipe produk, garansi produk dan foto utama produk pada halaman perbarui produk	PJ
SM_F_22	Hapus produk	Sistem harus mampu menyediakan fungsi hapus produk.	PJ

Kode fungsi	Nama Fungsi	Deskripsi/Spesifikasi	Pengguna
		Spesifikasi Kebutuhan: 1. Sistem harus mampu menampilkan tombol hapus produk pada setiap produk yang tersedia.	
SM_F_023	Tampil daftar produk penjual	Sistem harus mampu menyediakan fungsi tampil daftar produk penjual **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan halaman tampil daftar produk yang berisi nama produk, kategori produk status produk dan tombol *action* similarity grup facebook, tombol *action* hapus produk, tombol *action* edit produk dalam bentuk tabel.	PJ
SM_F_24	Bagikan info produk	Sistem harus mampu menyedikan fungsi bagikan info produk ke grup facebook **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan data lengkap dari produk yang berisi judul produk, kategori produk, berat produk, harga produk, jenis produk, deskripsi 2. Sistem mampu menampilkan daftar grup yang telah ditambahkan oleh akun.	PJ

Kode fungsi	Nama Fungsi	Deskripsi/Spesifikasi	Pengguna
		3. Sistem harus mampu menampilkan hasil perhitungan kemiripan antara produk dengan grup *facebook* sesuai dengan data *grup facebook* yang tersedia. 4. Sistem mampu menyediakan tombol checkbox untuk memilih *grup* 5. Sistem harus mampu menyediakan formulir keterangan yang befungsi untuk menambahkan keterangan pada saat produk dibagikan ke *grup facebook* 6. Sistem harus mampu menyediakan tombol *post facebook*..	
SM_F_25	Tambah *grup facebook*	Sistem mampu menyediakan fungsi tambah *grup facebook* **Spesifikasi Kebutuhan:** 1. Sistem mampu menyediakan formulir pencarian nama *grup* dan tombol pencarian 2. Sistem mampu menampilkan hasil pencarian yang berisi nama *grup*, id *grup*, dan jumlah member *grup* dalam bentuk tabel. 3. Sistem mampu menyediakan tombol tambah *grup* pada	PJ

Kode fungsi	Nama Fungsi	Deskripsi/Spesifikasi	Pengguna
		setiap *data grup* pada hasil pencarian.	
SM_F_26	Hapus *grup facebook*	Sistem harus mampu menyediakan fungsi hapus *grup facebook* **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan tombol hapus pada setiap *grup facebook* yang tersedia	PJ
SM_F_27	Konfirmasi transaksi produk Penjual	Sistem harus mampu menyediakan fungsi konfirmasi transaksi produk penjual **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan detail dari transaksi produk yang berisi , nama produk, jumlah, catatan transaksi, alamat tujuan, kodepos tujuan, total tagihan, kurir dan kode pembayaran. 2. Sistem harus mampu menampilkan tombol konifirmasi transaksi diterima atau transaksi ditolak.	PJ
SM_F_28	Tampil Profil member	Sistem harus mampu menyediakan fungsi tampil profil member. **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan halaman profil member yang berisi nama, username,	PB, PJ

Kode fungsi	Nama Fungsi	Deskripsi/Spesifikasi	Pengguna
		email, alamat, no *handphone,* catatan penjual, dan foto profil.	
SM_F_29	Perbarui Profil member	Sistem harus mampu menyediakan fungsi perbarui *profil* member **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan halaman perbarui *profil member* yang berisi nama, *username*, *email*, alamat, no *handphone*, catatan penjual, dan foto profil dalam bentuk *form*.	AD, PB, PJ
SM_F_30	tambah item keranjang Belanja	Sistem harus mampu menyediakan fungsi tambah keranjang belanja **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menyediakan tombol tambah item keranjang belanja beserta jumlah produk yang di tambahkan.	PB
SM_F_31	Lihat keranjang belanja	Sistem harus mampu menyediakan fungsi lihat keranjang belanja **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan daftar keranjang belanja yang berisi nama produk, jumlah produk, berat produk dan total produk dalam bentuk tabel.	PB

Kode fungsi	Nama Fungsi	Deskripsi/Spesifikasi	Pengguna
SM_F_32	Hapus *item* keranjang belanja	Sistem harus mampu menyediakan fungsi hapus *item* keranjang belanja **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan tombol hapus pada setiap item yang tersedia pada daftar keranjang belanja.	PB
SM_f_33	Transaksi jual	Sistem harus mampu menyediakan fungsi transaksi jual **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan halaman transaksi jual yang berisi produk yang terjual yang ditampilkan bedasarkan nama produk, waktu transaksi diterima.	PJ
SM_F_34	Riwayat belanja	Sistem harus mampu menyediakan fungsi riwayat belanja **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan halaman riwayat belanja yang berisi daftar produk yang terbeli yang ditampilkan bedasarkan nama produk, waktu transaksi.	PB

Kode fungsi	Nama Fungsi	Deskripsi/Spesifikasi	Pengguna
SM_F_35	Koneksi facebook	Sistem harus mampu menyediakan fungsi koneksi facebook **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan halaman koneksi *facebook* yang berisi tombol koneksi ke *facebok*, form nama depan, nama belakang, *email* dan alamat *url facebook*	PJ
SM_F_36	Similarity grup facebook	Sistem harus mampu menyediakan fungsi similarity grup *facebook* terhadap **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan data lengkap dari produk yang berisi judul produk, kategori produk, berat produk, harga produk, jenis produk, deskripsi 2. Sistem mampu menampilkan daftar grup yang telah ditambahkan oleh akun. 3. Sistem harus mampu menampilkan hasil perhitungan kemiripan antara produk dengan grup *facebook* sesuai dengan *data* grup *facebook* yang tersedia. 4. Sistem mampu menyediakan tombol	PJ

Kode fungsi	Nama Fungsi	Deskripsi/Spesifikasi	Pengguna
		checkbox untuk memilih grup 5. Sistem harus mampu menyediakan formulir keterangan yang befungsi untuk menambahkan keterangan pada saat produk dibagikan ke grup *facebook* 6. Sistem harus mampu menyediakan tombol *post facebook*.	
SM_F_37	*checkout*	Sistem harus dapat menyediakan fungsi *checkout.* **Spesifikasi Kebutuhan:** 1. Sistem harus dapat menampilkan halaman *checkout* yang berisi alamat tujuan, kurir, catatan transaksi, total transaksi dan metode pembayaran.	PB
SM_F_38	Tampil daftar alamat member	Sistem harus dapat menyediakan fungsi tampil daftar alamat. **Spesifikasi Kebutuhan:** 1. Sistem harus dapat menampilkan halaman daftar alamat yang berisi, nama alamat, provinsi,kota, kecamatan, kelurahan, jalan dan kodepos dalam bentuk tabel	PB, PJ

Kode fungsi	Nama Fungsi	Deskripsi/Spesifikasi	Pengguna
SM_F_39	Tambah alamat member	Sistem harus dapat menyediakan fungsi tambah alamat **Spesifikasi Kebutuhan:** 1. Sistem harus dapat menampilkan halaman tambah alamat yang berisi nama alamat, provinsi,kota, kecamatan, kelurahan, jalan dan kodepos dalam bentuk form.	PB, PJ
SM_F_40	Hapus alamat member	Sistem harus dapat menyediakan fungsi hapus alamat. **Spesifikasi Kebutuhan:** 1. Sistem harus dapat menyediakan tombol hapus alamat.	PB, PJ
SM_F_41	Tambah testimoni	Sistem harus dapat menyediakan fungsi tambah testimoni **Spesifikasi Kebutuhan:** 1. Sistem harus mampu menampilkan bagian tambah testimoni yang berisi *ratting* produk, dan pesan testimoni.	PB

4.2.3 Kebutuhan Non-Fungsional Sistem

Kebetuhan Non-fungsional adlah suatu analisis yang di gunakan untuk mengetahui kebutuhan perangkat lunak non-fungsional yang berhubungan dengan kualits sistem. Berikut kebutuhan non-fungsional yang di buat:

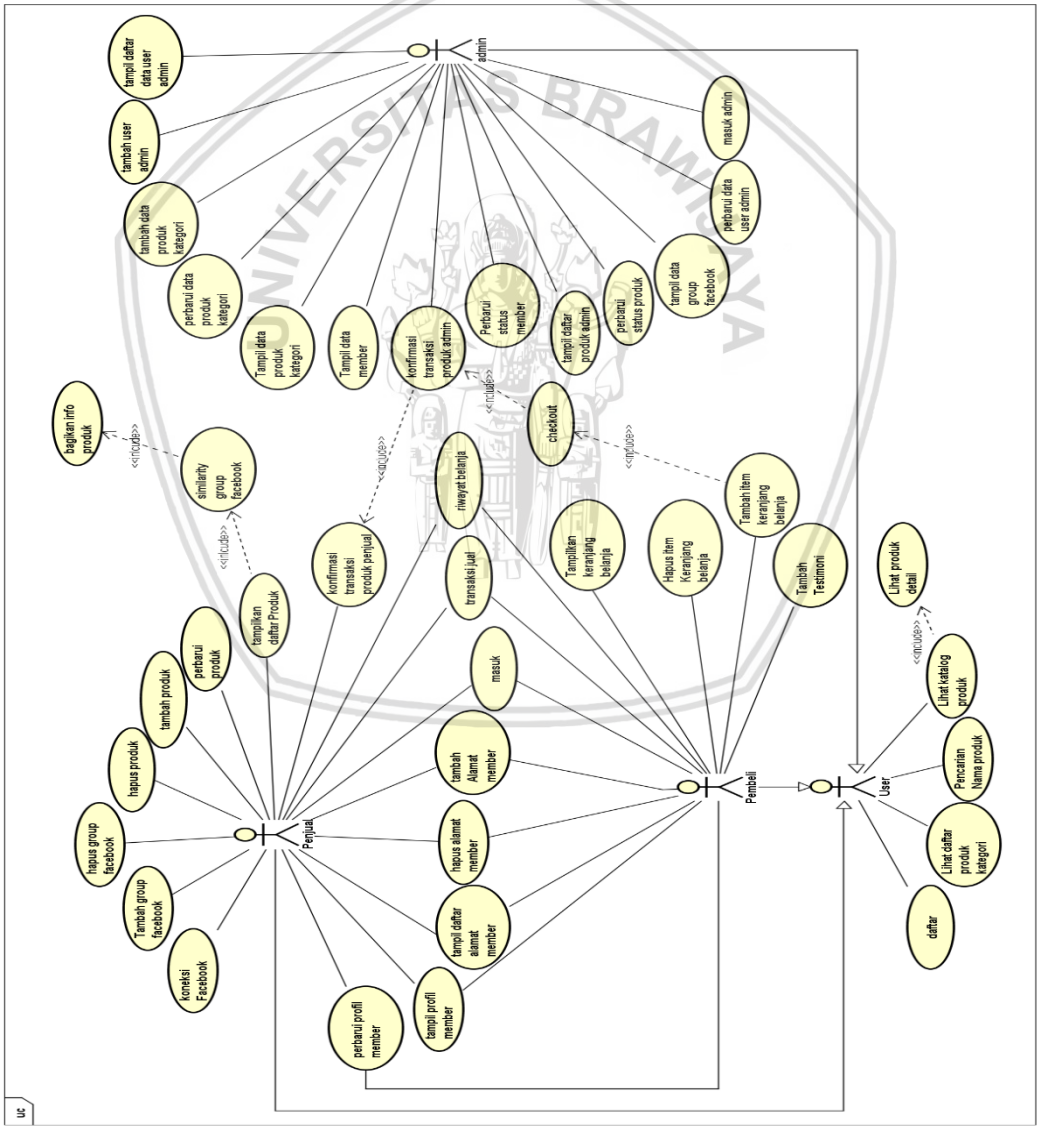
Tabel 4.4 Tabel kebutuhan non-fungsional

Kode fungsi	Parameter	Deskripsi
SM_NF_01	*Compatibility*	Sistem harus dapat dibuka pada browser *Mozilla Firefox*, *Microsoft edge*, *Chrome*.

4.3 Permodelan Analisis Kebutuhan

4.3.1 *Use Case* Diagram

Pada bagian ini menjelaskan tentang diagram *Use case. Use case* adalah diagram yang membahas tentang interaksi antara *user* pengguna dan *admin* dengan sistem *marketplace*.



Gambar 4.7 *Use Case Diagram*

Pada gambar 4.7 *Use case* menjelaskan tentang interaksi antara user pengguna dan juga admin terhadap sistem *marketplace*. Pada *use case* pertama yaitu menjalaskan tentang interaksi antara *user*, penjual, pembeli, *admin* dengan sistem *marketplace* dimana *user* pengguna untuk menampilkan kategori , daftar produk dan pencarian user pengguna tidak harus melakukan *login* terlebih dahulu, sedangkan untuk fitur lainya pembeli atau penjual diharuskan *login* terlebih dahulu, seperti fitur tambah produk, *update* produk dan lain sebagainya. Untuk bagian *admin* dengan sistem *marketplace*, dimana *admin* harus login terlebih dahulu untuk dapat menggunakan fitur dibagian *admin* panel, fitur tersebut meliputi tampil *data* kategori, *update* kategori, hapus kategori, tampilkan data user, *update* data user, tampil data produk, transaksi dan lain sebagainya.

4.3.2 Skenario Use case

Pada bagian Skenario *Use case* ini menjelaskan tentang uraian dari *use case* yang telah di jelaskan pada gambar sebelumnya. Pada skenario *use case* ini akan menjelaskan tentang aktor yang berkaitan dengan *use case*, Berikut *Skenario Use case* pada sistem marketplace:

1. *Skenario Use case* masuk

Tabel 4.5 *Skenario Use case* masuk admin

Skenario *Use case* **masuk admin**	
Objective	*Use case* ini mendeskripsikan admin untuk melakukan masuk ke dasborad sebagi admin
Actor	Admin
Pre-Condition	admin harus membuka web *marketplace* terlebih dahulu sebelum *use case* dilakukan kemudian masuk ke halaman masuk admin
Main Success Scenario	1. User membuka halaman *form* masuk admin 2. Sistem menampilkan *form* masuk admin 3. User memasukan *username* dan *password* pada *form* masuk admin dan klik tombol *login* 4. Sistem menampilkan *dasboard admin*
Alternative Flow	Jika data username dan password salah maka akan menampilkan pesan “username dan password Anda salah”
Post-Condition	Sistem menampilkan halaman *dasboard admin*

2. Skenario *Use case* tampil data member

Tabel 4.6 Skenario *Use case* tampil *data* member

Skenario *Use case* **tampil *data* member**	
Objective	*Use case* ini mendeskripsikan admin menampilkan seluruh member yang terdaftar
Actor	Admin
Pre-Condition	Sistem menampilkan halaman admin
Main Success Scenario	1. admin memilih *menu data member* 2. Sistem manampilkan daftar member yang berisi *username*, nama pengguna, status, tanggal daftar dan tombol *action detail member*
Alternative Flow	Jika data member masih kosong maka akan menampilkan pesan "member belum tersedia"
Post-Condition	Sistem menampilkan daftar penjual dan pembeli yang terdaftar

3. Skenario *Use case* perbarui *status* member

Tabel 4.7 Skenario *Use case* perbarui *status member*

Skenario *Use case* **perbarui *status member***	
Objective	*Use case* ini mendeskripsikan admin melakukan perbarui pada data member yang terdaftar
Actor	Admin
Pre-Condition	Sistem menampilkan halaman admin
Main Success Scenario	1. admin memilih *menu data member* 2. Sistem manampilkan daftar member 3. Admin memilih salah satu member kemudian tekan icon yang tedapat pada kolom *action*. 4. Sistem menampilkan detail member 5. Admin melakukan perubahan *status* member 6. Sistem menyimpan hasil perubahan yang dilakukan oleh admin
Alternative Flow	Jika data member masih belum tersedia maka akan menampilkan pesan "member belum tersedia"
Post-Condition	Sistem manpilkan detail member sesuai dengan yang telah admin perbarui

4. Skenario *Use case* tampil daftar produk *admin*

Tabel 4.8 Skenario *Use case* tampil daftar produk *admin*

Skenario *Use case* **tampil daftar produk *admin***	
Objective	*Use case* ini mendeskripsikan admin menampilkan seluruh produk yang tersedia
Actor	Admin
Pre-Condition	Sistem menampilkan halaman admin
Main Success Scenario	1. admin memilih *menu data* produk 2. Sistem manampilkan daftar produk yang berisi no, nama produk, kategori, *status,* tanggal post dan *action detail.*
Alternative Flow	Jika produk masih belum tersedia atau kosong maka akan menampilkan pesan"produk masih kosong"
Post-Condition	Sistem menampilkan daftar produk yang tersedia

5. Skenario *Use case* perbarui *status* produk

Tabel 4.9 Skenario *Use case* perbarui *status* produk

Skenario *Use case* **perbarui *status* produk**	
Objective	*Use case* ini mendeskripsikan admin melakukan perubahan pada data produk yang tersedia pada sistem
Actor	Admin
Pre-Condition	Sistem menampilkan halaman admin
Main Success Scenario	1. admin memilih *menu data* produk 2. Sistem manampilkan daftar produk yang tersedia 3. Admin memilih salah satu produk kemudian tekan icon yang tedapat pada kolom *action*. 4. Sistem menampilkan detail produk. 5. Admin melakukan perubahan *status* produk 6. Sistem menyimpan hasil perubahan yang dilakukan oleh admin
Alternative Flow	Apabila jaringan internet terputus saat melakukan perubahan status maka statatus dari data produk tidak akan berubah.

Post-Condition	Sistem manpilkan detail produk dengan hasil perubahan paling akhir

6. Skenario *Use case* tampil *data* kategori

Tabel 4.10 Skenario *Use case* tampil *data* produk kategori

Skenario *Use case* **tampil *data* produk kategori**	
Objective	*Use case* ini mendeskripsikan admin menampilkan data kategori produk yang tersedia
Actor	Admin
Pre-Condition	Sistem menampilkan halaman admin
Main Success Scenario	1. admin memilih *data* kategori produk 2. Sistem manampilkan daftar kategori produk yang berisi no, nama menu, dan tombol *action edit* dan juga form tambah kategori
Alternative Flow	Apabila data produk kategori masih kosong, maka akan memunculkan pesan "data produk kategori belum tersedia".
Post-Condition	Sistem menampilkan seluruh kategori produk yang tersedia

7. Skenario *Use case* tambah data kategori produk

Tabel 4.11 Skenario *Use case* tambah tambah data kategori

Skenario *Use case* **tambah *data* kategori produk**	
Objective	*Use case* ini mendeskripsikan admin melakukan penambahn *data* kategori produk baru
Actor	Admin
Pre-Condition	Sistem manampilkan halaman admin
Main Success Scenario	1. admin memilih *menu data* kategori produk 2. Sistem manampilkan daftar kategori produk yang berisi no, nama menu, dan tombol *action edit* dan juga *form* tambah kategori 3. Admin melakukan *input* data kategori produk baru pada *form* tambah kategori produk dan menekan tombol tambah.

	4. Sistem menyimpan data yang telah dimasukan oleh admin dan menampilkan pada daftar kategori produk.
Alternative Flow	Jika nama kategori masih kosong saat melakukan penambahan, maka akan menampilkan pesan "harap isi bidang ini"
Post-Condition	Sistem menampilkan daftar kategori produk yang tersedia pada halaman *data* kategori produk

8. Skenario *Use case* perbarui data kategori produk

Tabel 4.12 Skenario *Use case* perbarui data kategori produk

Skenario *Use case* **perbarui *data* kategori produk**	
Objective	*Use case* ini mendeskripsikan admin melakukan perubahan kategori produk
Actor	Admin
Pre-Condition	Sistem menampilkan halaman admin
Main Success Scenario	1. admin memilih *menu data* kategori produk 2. Sistem manampilkan daftar Kategori produk 3. Admin memilih salah satu kategori kemudian tekan *icon* pada bagian *action*. 4. Sistem mengubah nama kategori produk tersebut sesuai dengan kondisi kategori pada saat itu.. 5. Sistem akan menampilkan hasil perubahan pada daftar kategori produk yang tersedia
Alternative Flow	Apabila text pada *form* nama kategori dikosongkan maka akan muncul pesan "form tidak boleh kosong"
Post-Condition	Sistem manampilkan hasil dari perubahan kategori produk oleh admin

9. Skenario *Use case* Konfirmasi transaksi produk admin

Tabel 4.13 Skenario *Use case* konfirmasi admin

Skenario *Use case* **Konfirmasi *transaksi* produk *admin***	
Objective	*Use case* ini mendeskripsikan admin melakukan konfirmasi transaksi produk
Actor	Admin

Pre-Condition	Sistem menampilkan halaman admin
Main Success Scenario	1. admin memilih *menu data* penjualan 2. Sistem manampilkan daftar transaksi yang tersedia 3. Admin memilih salah satu data transaksi dari daftar transaksi yang akan dikonfirmasi. 4. Sistem menampilkan *detail* dari transaksi yang telah dipilih admin 5. Admin akan melakukan pengecekan transaksi. Apabila jika pembeli sudah melakan tranfer ke rekening admin, maka admin akan melakukan konfirmasi “terima”
Alternative Flow	Apabila saat konfirmasi masih transaksi produk jaringan *internet* terputus maka hasil konfirmasi tidak akan disimpan dalam *database*
Post-Condition	Sistem *data* transaksi yang telah dikonfirmasi oleh admin

10. Skenario *Use case* tampil data grup facebook

Tabel 4.14 Skenario *Use case* tampil data *grup facebook*

Skenario *Use case* **tampil *data grup facebook***	
Objective	*Use case* ini mendeskripsikan admin menampilkan seluruh *grup facebook* yang tersedia
Actor	Admin
Pre-Condition	Sistem menampilkan halaman admin
Main Success Scenario	1. admin memilih *menu data* grup fb 2. Sistem manampilkan daftar *grup facebook* yang berisi no, tag, nama grup, id grup, tanggal daftar dan tombol *action detail* grup
Alternative Flow	Jika data *grup facebook* masih kosong maka tabel akan menampilkan pesan ”data belum tersedia”
Post-Condition	Sistem menampilkan daftar *grup facebook* yang tersedia

11. Skenario *Use case* tampil daftar data user admin

Tabel 4.15 Skenario *Use case* tampil daftar data user admin

Skenario *Use case* **tampil daftar user admin**	
Objective	*Use case* ini mendeskripsikan admin menampilkan daftar *user admin* yang tersedia
Actor	Admin
Pre-Condition	Sistem menampilkan halaman admin
Main Success Scenario	1. admin memilih *menu user admin* 2. Sistem manampilkan daftar *user admin* yang berisi, no, username, nama, *status,* tombol *action* detail *user admindan* juga form tambah user admin
Alternative Flow	Apabila daftar *user admin* belum tersedia maka akan menampilkan pesan "*data user admin* kosong"
Post-Condition	Sistem menampilkan seluruh *user admin* yang tersedia

12. Skenario *Use case* tambah *user* admin

Tabel 4.16 Skenario *Use case* tambah *user admin*

Skenario *Use case* **tambah *user admin***	
Objective	*Use case* ini mendeskripsikan admin melakukan penambahn *user admin* baru
Actor	Admin
Pre-Condition	Sistem manampilkan halaman admin
Main Success Scenario	1. admin memilih *menu user admin* 2. Sistem manampilkan daftar *user admin* dan juga form tambah user admin 3. Admin melakukan *input* data user admin baru pada *form* tambah *user admin* dan menekan tombol tambah. 4. Sistem menyimpan *data* yang telah dimasukan oleh admin dan menampilkan pada daftar *user admin.*
Alternative Flow	1. Jika *form username* dan *password* belum diisi makan akan menampilkan pesan peringatan "harap diisi bidang ini"

	2. Jika *form* email diisi tidak sesuai dengan *form email* maka akan menampilkan pesan "sertakan @ pada alamat *email*"
Post-Condition	Sistem menampilkan daftar admin yang tersedia pada halaman *user admin*

13. Skenario *Use case* perbarui *data* user admin

Tabel 4.17 Skenario *Use case* perbarui *data user* admin

Skenario *Use case* **perbarui *data user admin***	
Objective	*Use case* ini mendeskripsikan admin melakukan perubahan data *user admin*
Actor	Admin
Pre-Condition	Sistem menampilkan halaman admin
Main Success Scenario	1. admin memilih *menu user admin* 2. Sistem manampilkan daftar *user admin* dan juga form tambah user admin 3. Admin memilih salah satu user admin kemudian tekan icon update. 4. Sistem menampilkan detail useradmin dan tombol aktif ataupun non aktif. 5. Admin melakukan perubahan *user* admin 6. Sistem menyimpan hasil perubahan yang dilakukan oleh admin
Alternative Flow	Apabila saat melakukan perubahan jaringan *internet* terputus maka hasil perubahan tidak akan disimpan dalam *database*
Post-Condition	Sistem manpilkan hasil dari perubahan data user admin

14. Skenario *Use case* daftar

Tabel 4.18 Skenario *Use case register*

Skenario *Use case* **Daftar**	
Objective	*Use* case ini mendeskripsikan user untuk melakukan daftar
Actor	User

Pre-Condition	User harus membuka web marketplace terlebih dahulu sebelum *use case* dilakukan kemudian masuk ke halaman daftar
Main Success Scenario	1. User membuka halaman utama 2. User memilih menu daftar 3. Sistem membuka halaman *form* daftar 4. User memasukan *data* diri pada *form* daftar dan menekan tombol daftar.
Alternative Flow	1. Jika data email sudah terdaftar menampilkan pesan"email sudah terdaftar" 2. Jika password tidak sama "password tidak sama"
Post-Condition	Sistem menampilkan pesan "daftar berhasil"

15. Skenario *Use case* masuk

Tabel 4.19 Skenario *Use case* masuk

Skenario *Use case* **masuk**	
Objective	*Use case* ini mendeskripsikan user untuk melakukan masuk ke sistem *marketplace*
Actor	Penjual, Pembeli
Pre-Condition	User harus membuka web marketplace
Main Success Scenario	1. Kemudian pilih *menu* masuk pada bagian kanan atas 2. Sistem membuka halaman *form* masuk 3. User mengisi data diri susui dengan yang ada pada *form* masuk
Alternative Flow	1. Jika data *username* dan *password* salah maka sistem akan menampilkan halaman *form* masuk kembali.
Post-Condition	Sistem menampilkan halaman profile dari user yang telah masuk

16. Skenario *Use case* Lihat katalog produk

Tabel 4.20 Skenario *Use case* melihat katalog produk

Skenario *Use case* **Lihat daftar Produk**	
Objective	*Use case* ini mendeskripsikan user pengguna untuk mengetahui katalog produk yang ada.

Actor	User
Pre-Condition	User pengguna harus membuka halaman sistem *marketplace*
Main Success Scenario	1. User membuka halaman katalog pada bagian halaman utama 2. Sistem menampilkan daftar produk yang ada di halaman katalog
Alternative Flow	1. Jika produk masih belum tersedia maka akan menampilkan pesan “produk kososng”
Post-Condition	Sistem menampilkan daftar produk pada halaman katalog

17. *Skenario Use case* Lihat detail produk

Tabel 4.21 *Skenario Use case* lihat detail produk

Skenario *Use case* **Lihat *detail* produk**	
Objective	*Use case* ini mendeskripsikan user pengguna untuk mengetahui detail produk yang ada.
Actor	User
Pre-Condition	Membuka halaman katalog pada bagian halaman utama
Main Success Scenario	1. Sistem menampilkan daftar produk 2. User memilih produk yang ingin dibuka detail 3. Sistem mengalihkan ke halaman detail produk yang dipilih user
Alternative Flow	-
Post-Condition	Sistem menampilkan detail produk yang telah dipilih user

18. Skenario *Use case* Lihat Daftar Produk Kategori

Tabel 4.22 Skenario *Use case* Lihat daftar produk kategori

Skenario Use case **Lihat daftar produk kategori**	
Objective	*Use case* ini mendeskripsikan user untuk melihat daftar kategori produk
Actor	User

Pre-Condition	User harus membuka web *marketplace*
Main Success Scenario	1. Pilih menu kategori 2. Kemudian sistem akan menampilkan daftar kategori pada menu bagian atas
Alternative Flow	Apa bila data produk belum tersedia maka akan menampilkan pesan "produk masih belum tersedia"
Post-Condition	Sistem menampilkan halaman daftar kategori produk pada halaman utama

19. Skenario *Use case* pencarian nama produk

Tabel 4.23 Skenario *Use case* Pencarian Nama Produk

Skenario *Use case* **Pencarian**	
Objective	*Use case* ini mendeskripsikan user pengguna untuk melakukan pencarian sesuai yang diinginkan
Actor	User
Pre-Condition	User harus membuka web marketplace
Main Success Scenario	1. Pilih form pencarian 2. Memasukan kata kunci yang ingin dicari pada form pencarian yang ada dihalaman utama 3. Sistem menampilkan hasil pencarian
Alternative Flow	1. Jika kata kunci tidak tersedia maka muncul pesan "kata kunci tidak tersedia"
Post-Condition	Sistem menampilkan hasil pencarian yang telah dimasukan oleh user pengguna

20. Skenario *Use case* tambah produk

Tabel 4.24 Skenario *Use case* tambah produk

Skenario *Use case* **Tambah produk**	
Objective	*Use case* ini mendeskripsikan penjual melakukan tambah produk
Actor	Penjual
Pre-Condition	Penjual masuk ke halaman dasboard member
Main Success Scenario	1. Penjual memilih menu tambah produk 2. Sistem menampilkan *form* tambah produk

	3. Penjual memasukan data produk yang ingin ditambahkan. Dan menekan tombol simpan. 4. Sistem melakukan penyimpanan data produk yang telah dimasukan.
Alternative Flow	1. Jika *form* judul,harga, berat, *qty* dan gambar belum terisi nama makan akan menampilkan pesan ”harap isi bagian ini”
Post-Condition	Sistem menampilkan daftar produk yang telah dimasukan.

21. Skenario *Use case* perbarui Produk

Tabel 4.25 Skenario *Use case perbarui* Produk

Skenario *Use case* **perbarui produk**	
Objective	*Use case* ini mendeskripsikan penjual melakukan perubahan produk
Actor	Penjual
Pre-Condition	Penjual masukan akun terlebih dahalu
Main Success Scenario	1. Penjual memilih menu semua produk 2. Sistem manampilkan daftar produk 3. Penjual memilih produk yang dirubah 4. sistem menampilkan detail produk yang dipilih 5. penjual memasukan data yang ingin dirubah. Dan menekan tombol simpan. 6. Sistem melakukan penyimpanan data yang telah dilakukan perubahan
Alternative Flow	1. Jika *form* judul,harga, berat, qty dan gambar belum terisi nama makan akan menampilkan pesan ”harap isi bagian ini”
Post-Condition	Sistem menampilkan daftar produk yang tersedia

22. Skenario *Use case* hapus Produk

Tabel 4.26 Skenario *Use case* hapus Produk

Skenario *Use case* **hapus produk**	
Objective	*Use case* ini mendeskripsikan penjual melakukan hapus produk

Actor	Penjual
Pre-Condition	Penjual masuk kehalaman tampil daftar produk
Main Success Scenario	1. Penjual memilih produk yang *diupdate* 2. sistem menampilkan detail produk yang dipilih 3. penjual memasukan data yang ingin diupdate. Dan menekan tombol update. 4. Sistem melakukan penyimpanan data yang telah dilakukan perubahan
Alternative Flow	Jika koneksi internet terputus atau tidak tersedia maka data produk tidak akan dihapus dari *database*
Post-Condition	Sistem menampilkan daftar produk yang tersedia

23. Skenario *Use case* Tampil daftar produk penjual

Tabel 4.27 Skenario *Use case* Tampilkan Daftar produk penjual

Skenario *Use case* **Tampil daftar produk penjual**	
Objective	*Use case* ini mendeskripsikan penjual manampilkan daftar produk yang telah dimasukan
Actor	Penjual
Pre-Condition	Penjual masuk kehalaman dasboard member
Main Success Scenario	1. Penjual memilih menu semua produk 2. Sistem menampilkan daftar produk yang berisi nama produk, kategori produk status produk ,tombol *action similarity,* tombol *action edit* produk dan tombol *action* hapus produk dalam bentuk tabel.
Alternative Flow	Jika data produk tidak tersedia maka sistem akan menampilkan pesan “produk belum tersedia”
Post-Condition	Sistem menampilkan dafta produk yang telah dinputkan oleh penjual

24. Skenario *Use case* bagikan info Produk

Tabel 4.28 Skenario *Use case* Bagikan *info* Produk

Skenario *Use case* **Bagikan *info* produk**

Objective	*Use case* ini mendeskripsikan penjual melakukan bagikan info produk ke *grup facebook*
Actor	Penjual
Pre-Condition	Penjual masuk halaman semua produk
Main Success Scenario	1. Penjual memilih produk dan menekan pada bagian *icon facebook* 2. Sistem merespon dengan meminta detail dari produk yang telah dipilih pada sistem 3. Kemudian sistem akan melakukan *request* data grup facebook bedasarkan id grup yang telah tersedia pada akun penjual tersebut. 4. Kemudian sistem akan melakukan perhitungan kemiripan antara produk yang dipilih oleh penjual dengan *grup facebook.* Data dari produk meliputi judul, deskripsi kategori dan subkategori. Dan pada data *grup facebook* meliputi judul grup, deskripsi, post terakhir digrup. Data produk dan data grup tersebut akan dilakukan perhitungan dengan menggunakan rumus *cosine similarity* yang telah ditentukan . 5. Sistem menampilkan hasil kemiripan data dari perhitungan dengan menggunakan rumus *cosine similarity* dan juga detail dari produk yang telah dipilih pada halaman bagikan *facebook*. 6. Penjual memilih grup yang ingin dibagikan info produk 7. Penjual mengisi form deskripsi untuk dibagikan 8. Penjual menekan tombol bagikan *facebook* 9. Sistem akan membagikan info produk ke grup yang dipilih
Alternative Flow	Apabila *data* grup belum ditambahkan maka tabel dari grup yang menghitung tingkat kemiripan tidak tersedia
Post-Condition	Info produk telah dibagikan ke grup *facebook* yang dinginkan

25. Skenario *Use case* tambah *grup facebook*

Tabel 4.29 Skenario *Use case* Tambah *Grup Facebook*

Skenario *Use case* **Tambah *grup facebook***	
Objective	*Use case* ini mendeskripsikan penjual melakukan tambah grup *facebook*
Actor	Penjual
Pre-Condition	Penjual masuk halaman grup *facebook*
Main Success Scenario	1. penjual pilih tombol tambah *grup facebook* 2. Sistem menampilkan *form grup facebook.* 3. Penjual memasukan data grup *facebook*, atau langsung mencari nama grup *facebook* pada form pencarian kemudian pilih tombol tambah. Dan tekan tombol simpan 4. Sistem akan menyimpan grup *facebook* yang telah ditambahkan
Alternative Flow	1. Sistem menampilkan pesan "grup *facebook* gagal disimpan"
Post-Condition	*Grup facebook* telah tersimpan pada *database* sistem

26. Skenario *Use case* hapus *grup facebook*

Tabel 4.30 Skenario *Use case* Hapus *grup facebook*

Skenario *Use case* **hapus *grup facebook***	
Objective	*Use case* ini mendeskripsikan penjual melakukan hapus grup facebook
Actor	Penjual
Pre-Condition	Penjual masuk kehalaman *grup facebook*
Main Success Scenario	1. Penjual memilih *grup facebook* yang ingin dihapus 2. Sistem melakukan penghapusan *data* grup *facebook* yang dipilih oleh penjual
Alternative Flow	Jika jaringan *internet* tidak tersedia atau terputus maka data *grup facebook* tidak akan dihapus dari *database*
Post-Condition	Sistem menampilkan data *grup facebook* yang tersedia

27. Skenario *Use case* Konfirmasi transaksi produk Penjual

Tabel 4.31 Skenario *Use case* Konfirmasi transaksi produk Penjual

Skenario *Use case* **Konfirmasi transaksi produk Penjual**	
Objective	*Use case* ini mendeskripsikan Konfirmasi transaksi produk Penjual
Actor	Penjual
Pre-Condition	Penjual masuk kehalaman transaksi jual
Main Success Scenario	1. Penjual memasuk halaman transaksi jual 2. Sistem menampilkan daftar transaksi baru 3. Penjual memilih transaksi yang ingin dikonfirmasi 4. Sistem menampilkan detail transaksi yang dipilih 5. Penjual menekan tombol "terima" atau "tolak"
Alternative Flow	Apabila tidak ada transaksi penjualan yang tersedia maka "transaksi kosong"
Post-Condition	Sistem menyimpan perubahan transaksi oleh penjual

28. Skenario *Use case* Tampil *Profile* member

Tabel 4.32 Skenario *Use case* Tampil *Profile* member

Skenario *Use case* ***Tampil Profile***	
Objective	*Use case* ini mendeskripsikan member melakukan tampil profil *member*
Actor	Penjual, Pembeli
Pre-Condition	member harus melakukan login terlebih dahulu untuk melakukan tampil profil *member*
Main Success Scenario	1. *Member* memilih menu bagian tampil profil 2. Sistem menampilkan *detail* profil dari *member*
Alternative Flow	Jika jaringan internet tidak tersedia atau terputus maka sistem tidak akan menampilkan *data profil*
Post-Condition	Sistem menampilkan detail profile member

29. Skenario *Use case* Perbarui *Profil* member

Tabel 4.33 Skenario *Use case* Perbarui *Profile* member

Skenario *Use case* **perbarui *Profile* member**

Objective	*Use case* ini mendeskripsikan *member* melakukan Perbarui profile
Actor	Penjual, Pembeli
Pre-Condition	Member harus melakukan login terlebih dahulu untuk melakukan perbarui *profile*
Main Success Scenario	1. Sistem menampilkan halaman dashboard member 2. Member mengarahkan ke menu perbarui *profile*. 3. Sistem menampilkan halaman perbarui *profile* yang terdapat *form* data diri yang dapat ubah. 4. Member mengisi *form* data diri yang ingin dirubah dan klik tombol simpan *profile*. 5. Sistem menyimpan perubahan *profile* yang telah dimasukan oleh member
Alternative Flow	Jika form email diisi tidak sesuai dengan format *email* maka akan menampilkan pesan "sertakan @ pada alamat email"
Post-Condition	Sistem menampilkan halaman profile yang telah dirubah oleh member

30. Skenario *Use case* tambah item Keranjang belanja

Tabel 4.34 Skenario *Use case* Tambah *Item* Keranjang belanja

Skenario *Use case* **Tambah *item* keranjang belanja**	
Objective	*Use case* ini mendeskripsikan pembeli melakukan tambah keranjang belanja
Actor	Pembeli
Pre-Condition	Pembeli harus melakukan login terlebih dahulu untuk melakukan tambah keranjang belanja
Main Success Scenario	1. pembeli memilih produk dan memilih tombol tambah keranjang belanja 2. sistem akan menambahkan produk yang dipilih pembeli ke keranjang belanja.
Alternative Flow	1. Jika stok produk 0 atau habis maka produk tidak bisa ditambahkan. 2. Jika penambahan item keranjang belanja melebihi batas stock produk maka akan menampilkan pesan

	"nulai harus lebih kecil atau sama dengan stock yang tersedia"
Post-Condition	Sistem berhasil menambkan produk ke keranjang belanja

31. Skenario *Use case* Tampilkan keranjang belanja

Tabel 4.35 Skenario *Use case* Tampilkan keranjang belanja

Skenario *Use case* **Lihat keranjang belanja**	
Objective	*Use case* ini mendeskripsikan pembeli melihat keranjang belanja
Actor	Pembeli
Pre-Condition	melakukan login terlebih dahulu untuk melihat keranjang belanja
Main Success Scenario	1. Pembeli memilih *icon* keranjang belanja pada menu bagian atas 2. Sistem menampilkan daftar produk yang sudah masuk pada keranjang belanja.
Alternative Flow	Jika tidak ada produk pada keranjang belanja maka akan muncul pesan"terdapat 0 keranjang belanja"
Post-Condition	Sistem menampilkan halaman keranjang belanja

32. Skenario *Use case* Hapus Item Keranjang belanja

Tabel 4.36 Skenario *Use case* Hapus Item Keranjang belanja

Skenario *Use case* **Hapus item *keranjang belanja***	
Objective	*Use case* ini mendeskripsikan pembeli melakukan hapus *item* keranjang belanja
Actor	Pembeli
Pre-Condition	Pembeli harus membeuka halaman keranjang belanja
Main Success Scenario	1. sistem menampilkan daftar item pada karanjang belanja 2. pembeli memilih salah satu item keranjang belanja yang ingin dihapus. 3. Sistem melakukan hapus pada item yang telah dipilih pembeli

Alternative Flow	Jika item keranjang belanja tidak tersedia pada keranjang belanja maka tidak ada item yang dihapus
Post-Condition	Sistem berhasil menghapus item pada keranjang belanja

33. Skenario *Use case* Transaksi jual

Tabel 4.37 Skenario *Use case* Transaksi jual

Skenario *Use case* T**ransaksi jual**	
Objective	*Use case* ini mendeskripsikan transaksi jual
Actor	Penjual, Pembeli
Pre-Condition	Penjual harus melakukan login ke akun masing-masing terlebih dahulu untuk melihat transaksi
Main Success Scenario	1. Member masuk kehalaman transaksi 2. Sistem menampilkan data transaksi yang tersedia 3. penjual memilih transaksi mana yang akan diproses. 4. Penjual menerima konfirmasi dari admin dan memasukan resi pengiriman produk ke sistem *marketplace* 5. Transaksi jual selesai
Alternative Flow	Apabila tidak ada transaksi penjualan yang tersedia maka “transaksi kosong”
Post-Condition	Sistem akan menampilkan keterangan pada setiap transaksi sesuai proses yang telah dikerjakan.

34. Skenario *Use case* riwayat belanja

Tabel 4.38 Skenario *Use case* Riwayat belanja

Skenario *Use case* Riwayat belanja	
Objective	*Use case* ini mendeskripsikan pembeli dalam melihat riwayat belanja
Actor	Pembeli, penjual
Pre-Condition	Member harus melakukan masuk ke akun masing-masing terlebih dahulu untuk melihat transaksi
Main Success Scenario	1. Pembeli masuk kehalaman riwayat belanja 2. Sistem menampilkan data transaksi

	3. Pembeli memilih transaksi mana yang akan diproses. 4. Sistem menampilkan detail dari transaksi yang dipilih 5. Pembeli akan memasukan bukti tranfer sesuai dengan nominal transaksaksi tesebut. 6. Sistem akan meneruskan transaksaksi keadmin dan dilanjutkan ke penjual untuk diproses 7. Pembeli menerima resi yang telah dimasukan oleh penjual.
Alternative Flow	Jika data riwayat belanja masih kosong maka akan menampilkan pesan "riwayat belanja belum tersedia"
Post-Condition	Sistem akan menampilkan keterangan pada setiap transaksi sesuai proses yang telah dikerjakan.

35. Skenario *Use case Koneksi Facebook*

Tabel 4.39 Skenario *Use case koneksi facebook*

Skenario Use case ***Koneksi Facebook***	
Objective	*Use case* ini mendeskripsikan penjual melakukan *koneksi* ke akun *facebook*
Actor	Penjual
Pre-Condition	Penjual harus melakukan login terlebih dahulu untuk melakukan koneksi *facebook*
Main Success Scenario	1. Penjual melakukan masuk akun terlebih dahulu 2. penjual pilih menu koneksi *facebook* 3. Sistem menampilkan halaman koneksi *facebook.* 4. Penjual melakukan koneksi ke akun *facebook* pribadi 5. Sistem sistem menyimpan data akun *facebook* penjual.
Alternative Flow	jIka *authentifikasi facebook* ditolak maka *koneksi facebook* tidak berhasil
Post-Condition	Sistem terkoneksi dengan akun *facebook* penjual

36. Skenario *Use case Similarity grup facebook*

Tabel 4.40 Skenario *Use case Similarity grup facebook*

Skenario *Use case* ***Similarity grup facebook***	
Objective	*Use case* ini mendeskripsikan penjual melakukan *Similarity* grup facebook
Actor	Penjual
Pre-Condition	Sistem menampilkan daftar produk yang tersedia
Main Success Scenario	1. Penjual memilih produk dan menekan pada bagian *icon facebook* 2. Sistem merespon dengan meminta detail dari produk yang telah dipilih pada sistem 3. Kemudian sistem akan melakukan *request* data *grup facebook* bedasarkan id grup yang telah tersedia pada akun penjual tersebut. 4. Kemudian sistem akan melakukan perhitungan kemiripan antara produk yang dipilih oleh penjual dengan *grup facebook.* Data dari produk meliputi judul, deskripsi kategori dan subkategori. Dan pada data *grup facebook* meliputi judul grup, deskripsi, post terakhir pada grup. Data produk dan data grup tersebut akan dilakukan perhitungan dengan menggunakan rumus *cosine similarity* yang telah ditentukan . 5. Sistem menampilkan hasil kemiripan data dari perhitungan dengan menggunakan rumus *cosine similarity* dan juga detail dari produk yang telah dipilih pada halaman bagikan *facebook*. 6. Penjual memilih grup yang ingin dibagikan *info* produk 7. Penjual mengisi form deskripsi untuk dibagikan 8. Penjual menekan tombol bagikan *facebook* 9. Sistem akan membagikan info produk ke *grup* yang dipilih
Alternative Flow	Jika data grup tidak tersedia maka sistem akan menampilkan pesan "*grup facebook* tidak tersedia"
Post-Condition	Hasil kemiripan antara grup facebook dan produk ditampilkan dalam bentuk tabel

37. Skenario *Use case checkout*

Tabel 4.41 Skenario *Use case checkout*

Skenario *Use case* **checkout**	
Objective	*Use case* ini mendeskripsikan pembeli melakukan *checkout* produk
Actor	Pembeli
Pre-Condition	Penjual dan Pembeli harus melakukan login terlebih dahulu untuk melakukan *checkout* produk
Main Success Scenario	1. Pembeli masuk ke halaman keranjang belanja 2. Pembeli memilih produk yang ada pada keranjang belanja untuk dilakukan checkout 3. Sistem menampilkan detail produk yang dipilih oleh pembeli. 4. Pembeli menentukan alamat tujuan pengiriman beserta ongkos kirimnya. Dan menekan tombol submit 5. Sistem akan memindah produk kebagian transaksi jual untuk diproses selanjutnya
Alternative Flow	Jika produk tidak tersedia atau jumlah produk kosong maka akan menampilkan pesan “checkout gagal”
Post-Condition	Sistem memindah produk yang ada pada keranjang belanja ke transaksi untuk diproses selanjutnya

38. Skenario *Use case* Tampil daftar alamat member

Tabel 4.42 Skenario *Use case* Tampil daftar alamat member

Skenario *Use case* **Tampil daftar alamat**	
Objective	*Use case* ini mendeskripsikan member untuk melihat daftar alamat pada akun.
Actor	Penjual, pembeli
Pre-Condition	Member harus melakukan login terlebih dahulu untuk melihat daftar alamat
Main Success Scenario	1. member membuka halaman dasboard 2. Member memilih menu alamat

	3. Sistem menampilkan daftar alamat yang terdaftar sesuai dengan member yang login
Alternative Flow	Apabila data alamat tidak tersedia maka sistem akan menampilkan pesan " alamat tidak tersedia"
Post-Condition	Sistem menampilkan halaman daftar alamat

39. Skenario *Use case* Tambah alamat member

Tabel 4.43 Skenario *Use case* Tambah alamat

Skenario *Use case* **Tambah alamat**	
Objective	*Use case* ini mendeskripsikan member untuk melakukan tambah alamat yang dinginkan
Actor	Penjual, pembeli
Pre-Condition	Member harus melakukan login terlebih dahulu untuk melihat daftar alamat
Main Success Scenario	1. User membuka halaman dasboard 2. Penjual atau pembeli memilih menu alamat 3. Sistem menampilkan halaman daftar alamat 4. Penjual atau pembeli memilih tombol tambah alamat 5. Sistem menampilkan form tambah alamat yang sudah terdapat beberapa form alamat. 6. Penjual atau pembeli mengisi form tambah alamat kemudian klik tombol submit 7. Sistem akan menyimpan alamat yang ditambahkan oleh penjual atau pembeli
Alternative Flow	Jika ada bagian form yang belum terisi maka akan menampilkan pesan "harap isi bidang ini"
Post-Condition	Sistem menampilkan halaman daftar kategori pada halaman utama

40. Skenario *Use case* Hapus alamat member

Tabel 4.44 Skenario *Use case* Hapus alamat member

Skenario *Use case* **Hapus alamat**	
Objective	*Use case* ini mendeskripsikan member hapus alamat

Actor	Pembeli, penjual
Pre-Condition	Member harus melakukan login terlebih dahulu untuk melihat daftar alamat
Main Success Scenario	1. member memilih menu alamat 2. Sistem manampilkan daftar alamat yang tersedia 3. Member memilih alamat yang ingin dihapus dari sistem 4. Sistem menghapus alamat sesuai dengan pilihan member
Alternative Flow	Jika koneksi internet tidak tersedia maka data alamat tidak akan terhapus dari *database*
Post-Condition	Sistem berhasil menghapus alamat member

41. *Use case* Tambah Testimoni

Tabel 4.45 Skenario *Use case* Tambah Testimoni

Skenario *Use case* **Tambah Testimoni**	
Objective	*Use case* ini mendeskripsikan pembeli menambahkan testimoni
Actor	Pembeli
Pre-Condition	Pembeli melakukan login terlebih dahulu untuk menambahkan testimoni
Main Success Scenario	1. Pembeli memilih transaksi yang sudah selesai untuk ditambahkan testimoni 2. Sistem manampilkan detail transaksi beserta form testimoni 3. Pembeli mengisi form testimoni dan melakukan submit 4. Sistem menyimpan testimoni yang telah diinputkan pembeli
Alternative Flow	Jika koneksi *internet* tidak tersedia maka data testimoni tidak akan tersimpan dalam *database*
Post-Condition	Sistem berhasil menambahkan testimoni baru

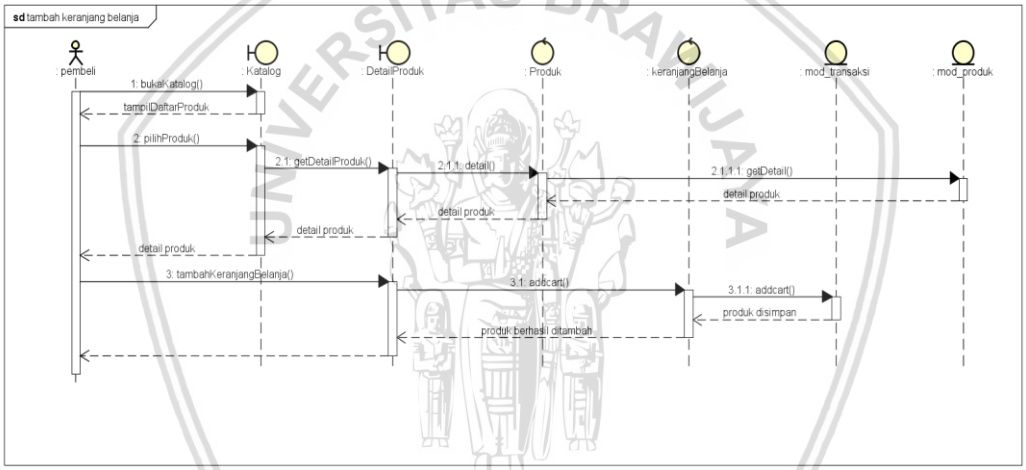
4.4 Perancangan Sistem

4.4.1 *Sequence Diagram*

Perancangan sequence diagram merupakan perwujudan dari penggambaran tingkah laku objek pada *use case diagram* dengan menjelaskan *life time* dari suatu objek, pesan yang dikirim dan diterima oleh antar objek. Jumlah *sequence diagram* yang dibuat adalah sebanyak *use case diagram* yang telah didefinisikan. Berikut *seqence diagram* yang telah dibuat.

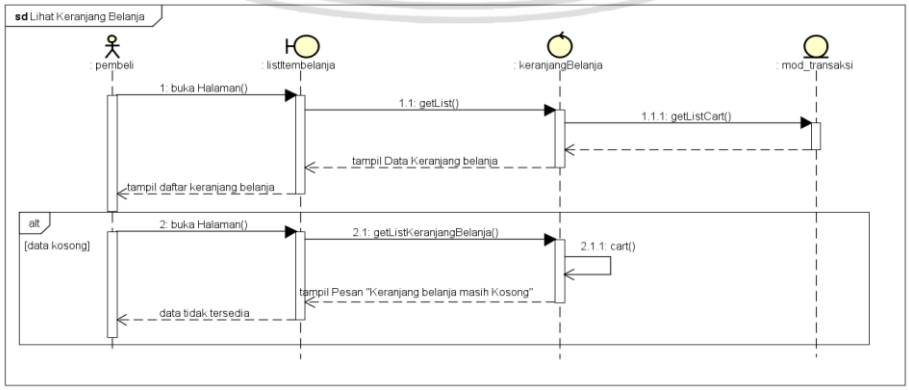
1. Tambah keranjang belanja atau fungsi *addcart*

   Pada gambar 4.8 dibawah menjelaskan interaksi antar objek dari fungsi tambah keranjang belanja. Interaksi dimulai dari pembeli melaukan pemilihan produk kemudin sistem merespon dengan menampilkan detail dari produk yang dipilih oleh pembeli, setelah detail produk tampil pembeli dapat melakukan penambahan produk kemudian data dari produk tersebut akan langsung menambah ke daftar keranjang belanja.



Gambar 4.8 *Sequence diagram* keranjang belanja

2. Lihat kerajang Belanja

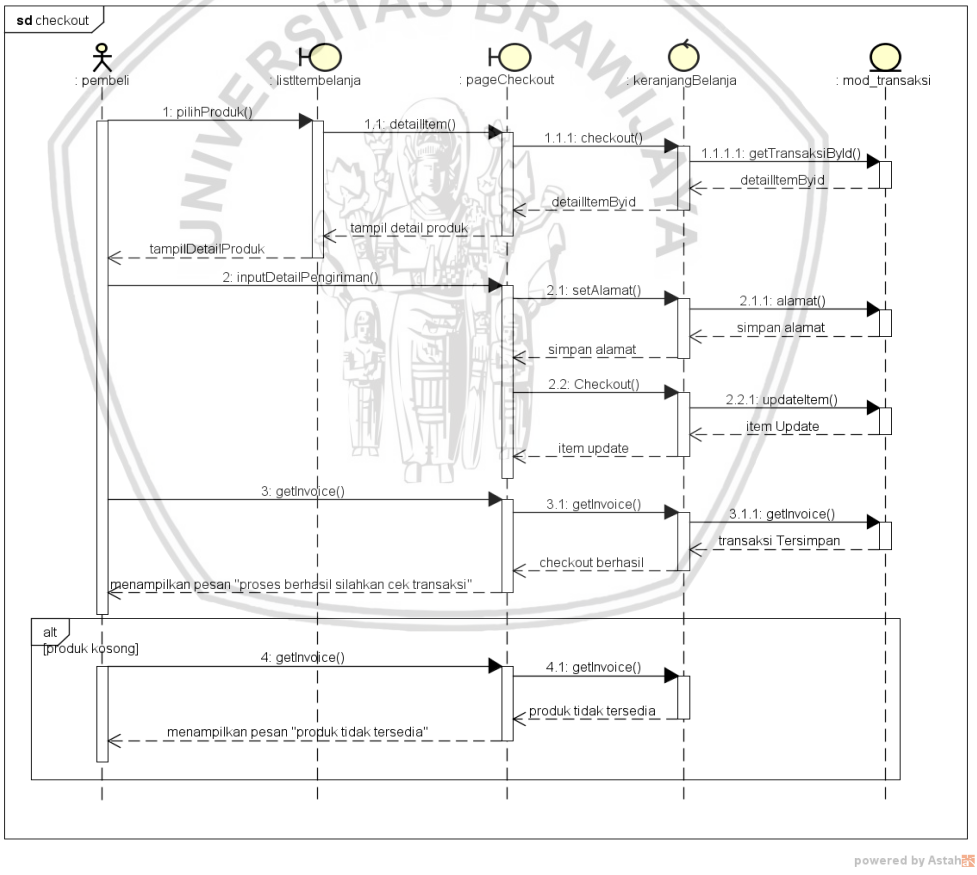


Gambar 4.9 *Sequence diagram* lihat keranjang belanja

Pada gambar 4.9 diatas menjelaskan tentang interaksi antar objek dari fungsi lihat keranjang belanja. Interaksi dimulai dari pembeli membuka halaman keranjang belanja. Kemudian data keranjang belanja diambil sesuai dengan data member yang *login* pada saat itu.

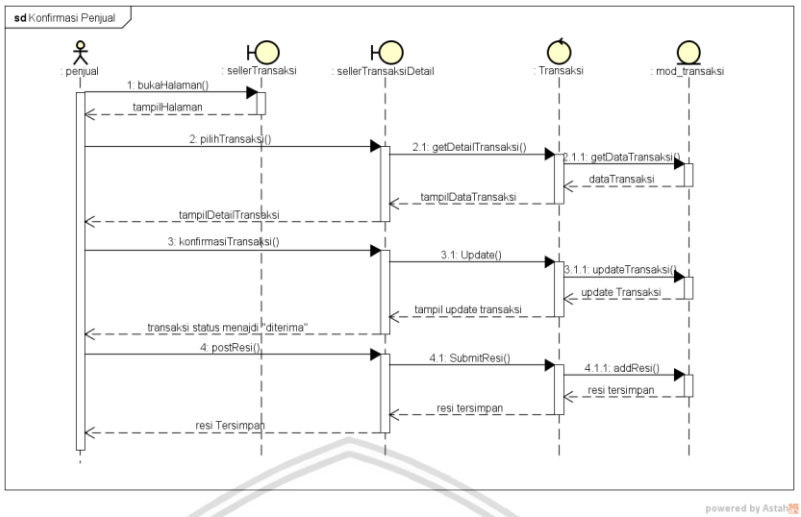
3. *Checkout*

Pada gambar 4.10 menjelaskan tentang intraksi antar objek dari fungsi checkout. Interaksi dimulai dari pembeli memilih produk yang sudah masuk dikeranjang belanja. Kemudian sistem akan merespon dengan menampilkan detail checkout pada saat menampilkan detail checkout pembeli meinputkan alamat pengiriman kemudian sistem akan langsung manampilkan ongkos kirim sesusi tujuan pengiriman. Pada proses submit checkout kelas checkout akan menggunakan fungsi tambah transaksi dan kemudian akan diteruskan oleh fungsi simpanTransaksi dimana fungsi tersebut akan memproses data checkout menjadi data transaksi yang akan dikirimkan ke penjual dan admin sistem.



Gambar 4.10 *Sequence diagram Checkout*

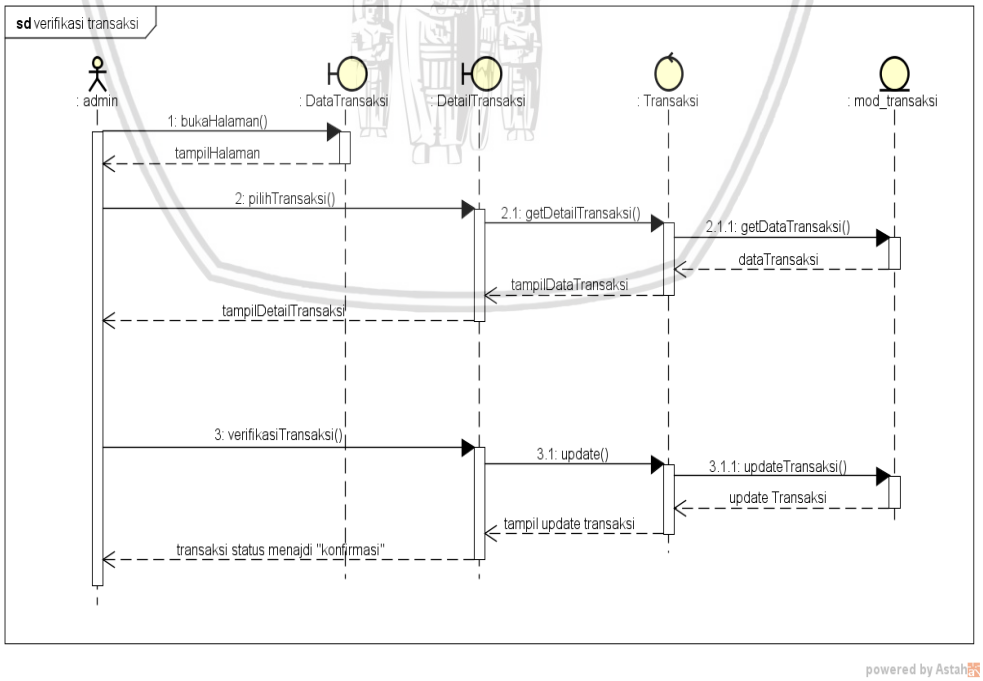
4. Konfirmasi Transaksi transaksi produk Penjual



Gambar 4.11 *Sequence diagram* Konfirmasi transaksi produk Penjual

Pada Gambar 4.11 menjelaskan tentang interaksi antar objek pada fungsi konfirmasi transaksi transaksi produk penjual. Dimana pada fungsi tersebut menjelaskan tentan proses manampilkan transaksi dan juga proses menampilkan detail transaksi sesuai dengan penjual, kemudian proses konfirmasi transaksi oleh penjual.

5. konfirmasi transaksi produk admin



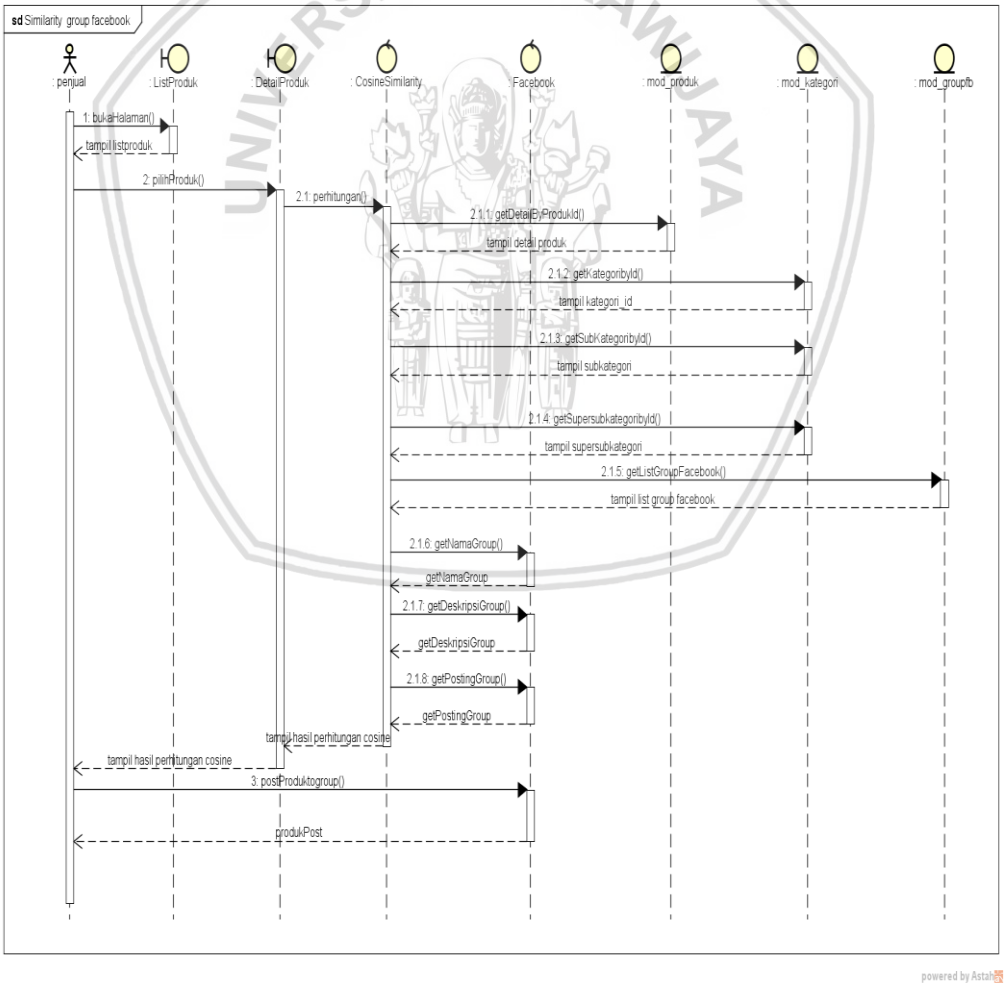
Gambar 4.12 *Sequence diagram* konfirmasi transaksi produk admin

Pada gambar 4.11 menjelaskan tentang interaksi antar objek pada fungsi konfirmasi transaksi produk admin. Dimulai dari admin membuka detail transaksi

yang akan diverifikasi, kemudian admin melakukan verifikasi dan akan direspon oleh fungsi update pada kelas transaksi, kemudian update transaksi akan disimpan pada database dengan menggunakan fungsi updatetransaksi.

6. *Sequence Similarity Grup* dan post Facebook

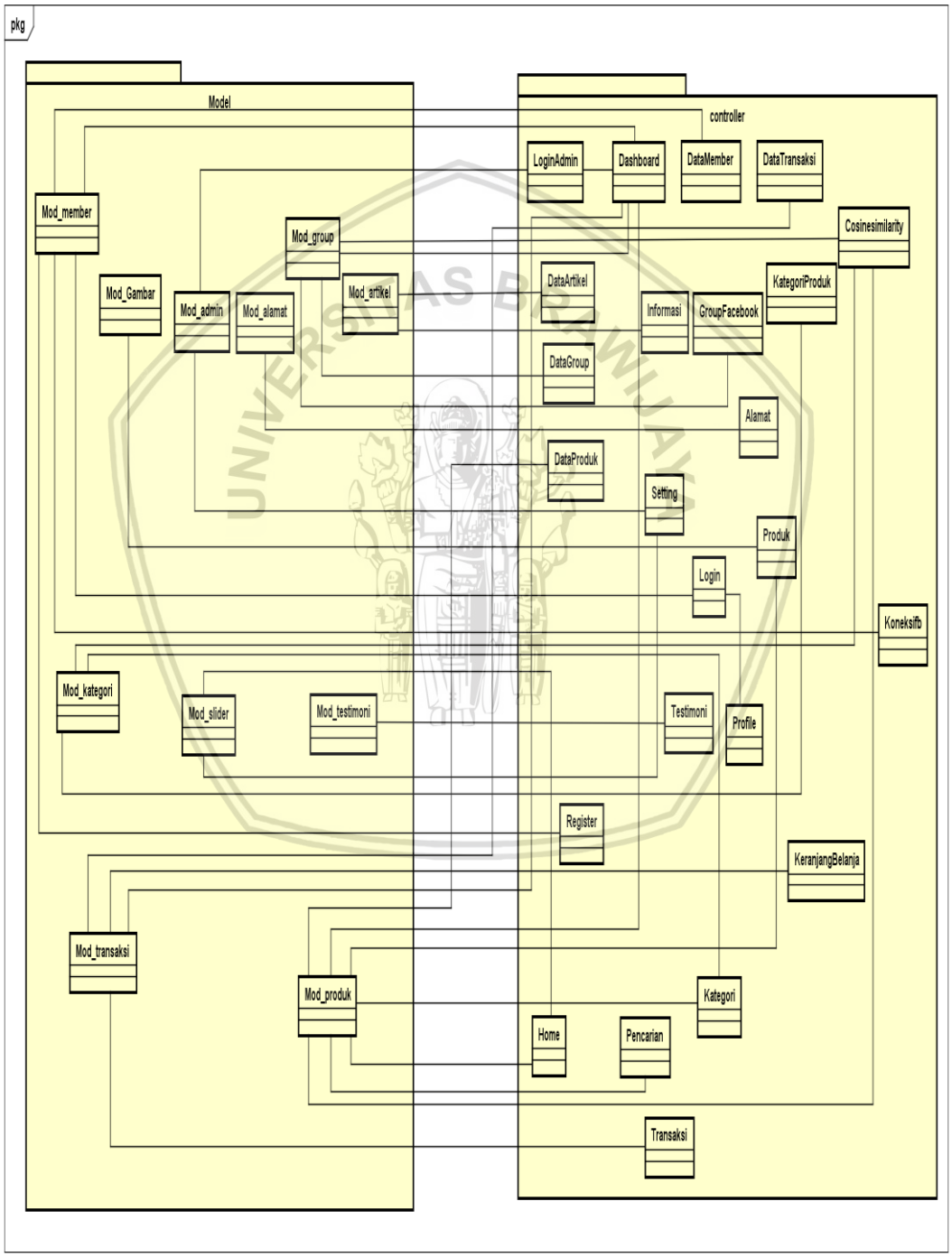
Pada gambar 4.13 menjelaskan tentang interaksi antar objek pada fungsi *similarity* grup *facebook*. Dimulai dari penjual memilih produk yang akan dibandingkan kemudian sistem akan merespon dengan menampilkan detail produk menggunakan fungsi perhitungan. Pada saat fungsi perhitungan dijalankan, sistem juga akan akan melakukan pemanggilan data produk yang dibutuhkan seperti data nama produk, deskripsi produk, kategori produk, subkategori produk dan juga sistem juga akan melakukan pemanggilan data grup *facebook* yang tersedia pada *account member* tersebut, pemanggilan data grup meliputi nama grup, deskripsi grup dan postingan paling akhir dari grup tersebut. Dari hasil pemanggilan data produk dan data grup tersebut kemudian akan dilakukan perhitungan tingkat kemiripan dan hasilnya akan ditampilkan pada halaman detail.



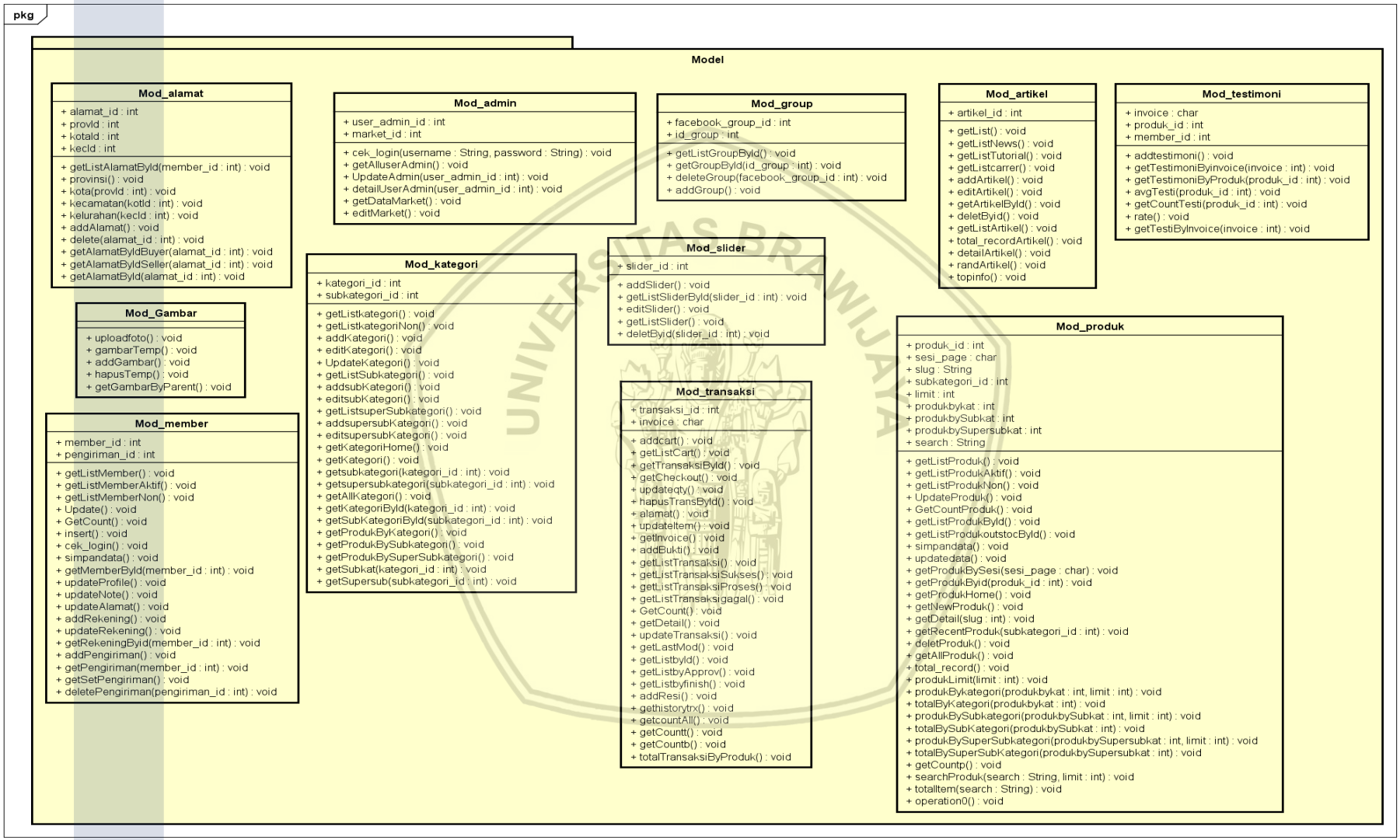
Gambar 4.13 *Sequence diagram* Similarity grup Facebook

4.4.2 Class Diagram

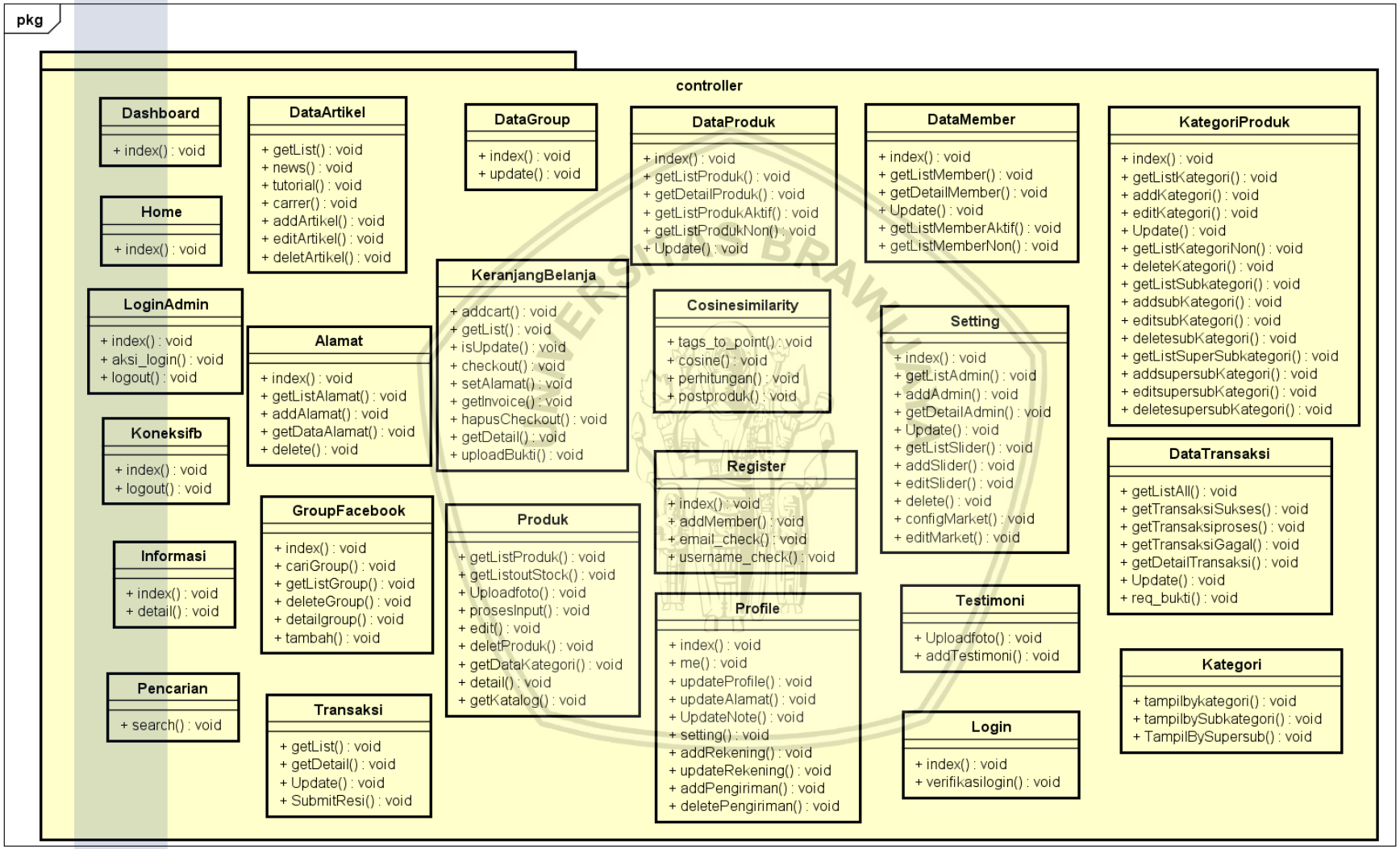
Pada bagian ini menjelaskan tentang perancangan dari diagram kelas yang bertujuan untuk menggambarkan sistem. Dimana hasil dari perancangan diagram tersebut menunjukkan bahwa setiap kelas controller akan memiliki hubungan *generalisasi* dengan ci_controller, begitu juga dengan setiap model akan memiliki hubungan generalisasi pada ci_model. Perancangan diagram kelas yang dilakukan menghasilkan diagram seperti pada gambar 4.14



Gambar 4.14 *Class diagram*



Gambar 4.15 *Class Diagram package model*



Gambar 4.16 *Class Diagram package controller*

4.4.3 Perancangan *Algoritme*

Perancangan *algoritme* ini merupakan perancangan kode program untuk diimplementasikan. Pada perancangan algoritme ini akan menggunakan pseudocode sebagai penjelasan dari algoritme yang diterapkan.

1. Perancangan pseudocode untuk koneksi facebook
   - Nama klas : Koneksifb.php
   - Nama operasi : index ()

Tabel 4.46 *Pseudocode* fungsi koneksi *facebook*

```
appID = ambil data app facebook developer
appSecret = ambil data app secret facebook developer
accesToken = appId appSecret
IF(facebook = is_authenticated) THEN
    ambil data first_name, last_name, email, gender, picture
    data masukan member_id  = set data session member login
    data masukan first_name = facebook first_name
    data masukan last_name  = facebook last_name
    data masukan email      = facebook email
    data masukan gender     = facebook gender
    data masukan picture    = facebook picture url
    datauser = simpan data member_id, first_name, last_name, email,
    gender, picture
    IF(dataUser !=null) THEN
        ambil data masukan member_id, first_name, last_name,
        email, gender, picture
        set session member_id, first_name, last_name, email, gender,
        picture
    ELSE THEN
        menampilkan halaman koneksifb
        menampilkan tombol logout
    END IF
ELSE THEN
    menampilkan tombol koneksi facebook
END IF
```

2. Perancangan pseudocode untuk *cosine similarity*

   - Nama klas : Cosinesimilarity.php
   - Nama operasi : perhitungan ()

Tabel 4.47 *Pseudocode* fungsi *cosine similarity*

```
id = id_produk
produk = id_produk
idkategori = kategori produk
idsubkategori = subkategori set idkategori
kata_produk = produk + kategori + subkategori
pisah_produk = explode(kata_produk)
member_id = set data member_id dengan session
grup = grupfb(session bedasarkan member_id)
foreach (grup)
    grupfb = mengambil data facebook nama,deskripsi, feed
    IF (!empty = feed) THEN
      foreach (feed)
         textjual = ambil data postingan grup
      End foreach
    END IF
    kata_grup = grupfb[name]+grupfb(description)+textjual
    pisah_grup = explode(kata_grup)
 End foreach
  FOR
   grupfb = data request facebook name
   articles id_grupfb = grup [id_grup]
   articles nama_grup = grupfb[name]
   articles tags      = pisah_grup
  END FOR
  target = set data pisah_grup
  tags = set array articles
  compare =  set target+ array articles tags
  foreach (articles)
```

```
   ak = set arrray articles tagss + tags
   idgrup = set data articles id_grup
   nama    = set data articles nama_grup
   score   = set data decimal ((katda_produk*kata_grup) kata_produk +
     kata_grup)/(kata_produk * kata_grup)/100
   end foreach
tampilkan halaman cosinesimilarity
```

3. Perancangan pseudocode untuk *post facebook*
   - Nama klas : Cosinesimilarity.php
   - Nama operasi : postproduk ()

Tabel 4.48 *Pseudocode* fungsi *post facebook*

```
checkbox = mengambil input sesuai checkbox
foreach (checkbox)
   ambil data detail grup
   post date ke grup facebook
end foreach
tampil halaman daftar produk
```

4. Perancangan pseudocode untuk menambahkan keranjang belanja
   - Nama klas : keranjangbelanja.php
   - Nama operasi : addcart()

Tabel 4.49 *Pseudocode* fungsi *addcart*

```
Slug = mengambil slug dari produk yang ditampilkan
IF(submit) THEN
    Input  data Qty
    Input data Seller
    Input data Member_Id
    Input data Gambar
    Input data Gambar
    Input data Status
    Simpan data input
```

```
    Tampil halaman detail produk detail by slug
ELSE THEN
     Tampil halaman detail produk detail by slug
END IF
```

5. Perancangan *pseudocode* untuk *checkout*
   - Nama klas : keranjangbelanja.php
   - Nama operasi : checkout ()

Tabel 4.50 *Pseudocode* fungsi *checkout*

```
Transaksi_id = mengambil data input transaksi_id
IF(submit) THEN
    Input Data Kurir
    Input Data Catatan
    Input data Ongkir
    Simpan input data
    Menampilkan halaman keranjangbelanja/checkout/transaksi_id
ELSE THEN
    Transaksi_id= data dari uri segment(3)
    Tampil data transaksi by transaksi_id
    Member_id = nilai member_id dari session
    Tampil data Pembeli by session
    Tampil alamat pembeli
    Tampil alamat tujuan
    Produk_id = detail transaksi bagian produk_id
    Tampil detail produk
    Member_id = data member_id bedasarkan produk
    Tampil detail penjual
    Tampil data pengiriman
    Tampil alamat penjual by penjual_id
    Tampil data alamat penjual
END IF
```

6. Perancangan pseudocode untuk dapatkan kode tagihan pembayaran
   - Nama klas : KeranjangBelanja.php
   - Nama operasi : getInvoice ()

Tabel 4.51 *Pseudocode* fungsi dapatkan kode tagihan pembayaran

```
IF(submit) THEN
    Transaksi_id = input transaksi_id
    Data transaksi = ambil data transaksi bedasarkan transaksi_id
    Produk_id = ambil produk_id bedasarkan data transaksi
    Data produk = ambil data produk bedasarkan produk_id
    IF (produk qty <= transaksi qty )
        Simpan transaksi
        Tampil data transaksi
        Tampil data pembeli
        Tampil data alamat
        Tampil data penjual
        Tampil alamat penjual
        Menampilkan halaman getInvoice
    ELSE THEN
        Menampilkan halaman soldout
    END IF
ELSE THEN
    Mengalihkan ke halaman checkout bedasarkan transaksi_id
END IF
```

7. Perancangan *pseudocode* untuk konfirmasi transaksi produk admin
   - Nama klas : dataTransaksi.php
   - Nama operasi : update ()

Tabel 4.52 *Pseudocode* fungsi konfirmasi transaksi produk admin

```
Update = false
Switch(action)
  Case 1
      Update =true
```

```
        Input Status = 1
        Input Time_approve = date
        Update data transaksi by id
        Tampil halaman Detail transaksi
    Case 2
        Update =true
        Input Status =0
        Update data transaksi by id
        Tampil halaman Detail transaksi
IF (update = false) THEN
        Tampil halaman daftar data transaksi
END IF
```

8. Perancangan *pseudocode* untuk konfirmasi transaksi produk penjual

- Nama klas : Transaksi.php
- Nama operasi : *update* ()

Tabel 4.53 *Pseudocode* fungsi konfirmasi transaksi produk penjual

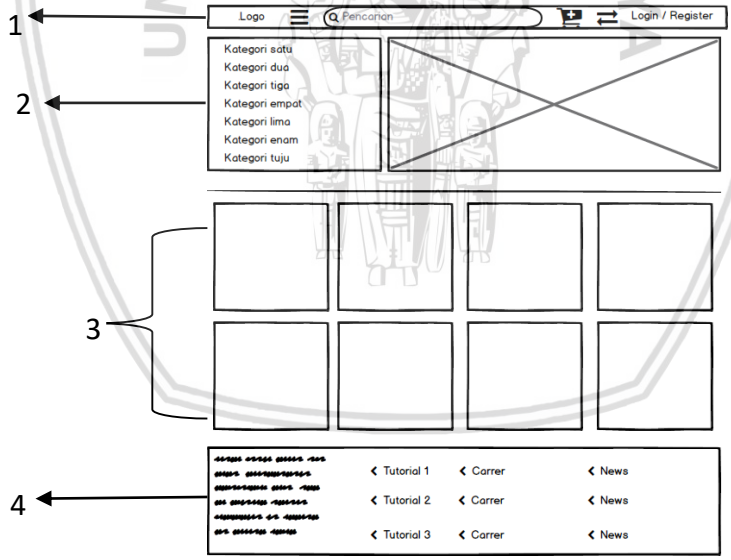
```
Update = false
Switch(action)
    Case 1
        Update =true
        Input Status = 2
        Input Time_approveseller = date
        Update data transaksi by id
        Tampil halaman Detail transaksi
    Case 2
        Update =true
        Input Status =1
        Update data transaksi by id
        Tampil halaman Detail transaksi
    Case 3
        Update =true
```

```
        Input Status =5
        Update data transaksi by id
        Tampil halaman Detail transaksi
    Case 4
        Update =true
        Input Status =1
        Update data transaksi by id
        Tampil halaman Detail transaksi
IF (update = false) THEN
        Tampil halaman daftar data transaksi
END IF
```

4.4.4 Perancangan Antarmuka

Perancangan antarmuka digunakan untuk memberikan gambaran dari tampilan sistem dengan bentuk kotak disertai dengan nomor dan penjelasannya.

1. Perancangan antarmuka halaman utama atau beranda

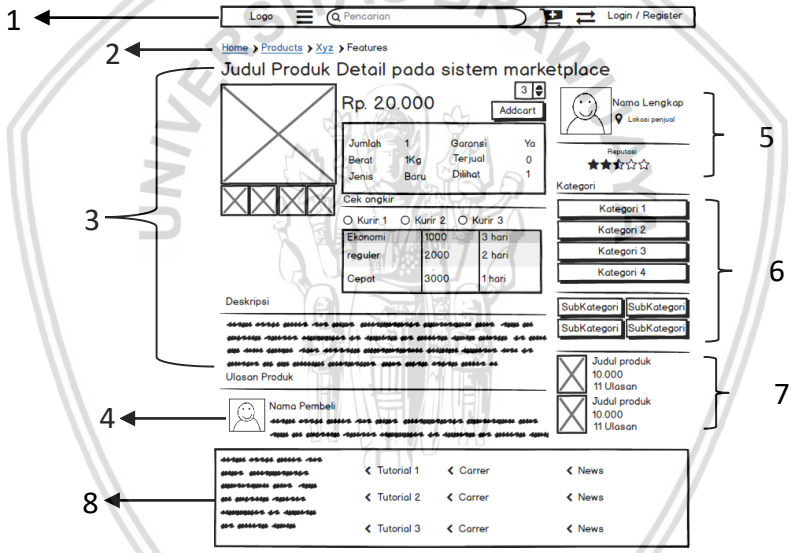


Gambar 4.17 Perancangan antarmuka halaman Utama atau beranda

Pada gambar 4.17 percangan antarmuka halaman utama atau beranda terdapat 4 nomor, nomor 1 merupakan bagian header yang berisi logo, *icon* kategori, *form* pencarian, *icon* belanja, icon transaksi diterima dan menu masuk atau login. Pada bagian nomor 2 terdapat kategori dan juga *slider* gambar pada sebelah kanan, pada bagian nomor 3 merupakan bagian produk, sebagian produk akan ditampilkan pada bagian 3 tersebut dan bagian 4 merupakan footer, footer berisi deskripsi dari sistem pada sebelah kiri, kemudian beberapa menu pilihan pada sebelah kanan.

2. Percangan antarmuka halaman *Detail* produk

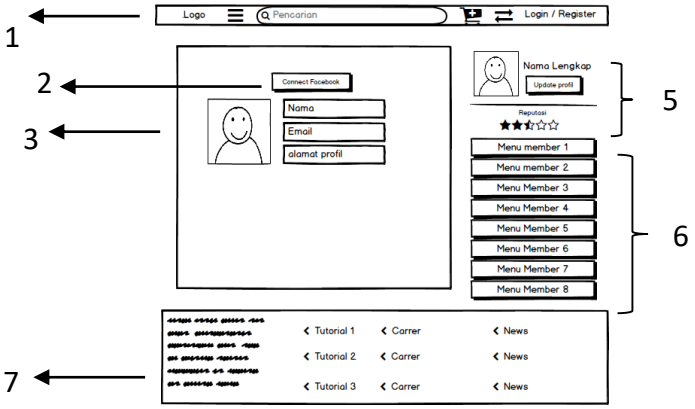
Perancangan antarmuka halaman detail produk pada gambar 4.18, pada gambar tersebut terdapat beberapa bagian. Bagian nomor 1 merupakan *header*, nomor 2 merupakan *navigasi* dari produk yang dibuka atau lebih sering disebut dengan *breadcrumb,* bagian nomor 3 merupakan detail dari produk yang terdiri dari judul produk, harga produk, gambar produk, jumlah produk, berat produk, total terjual transaksi tersebut dan juga deskripsi dari produk tersebut. Pada bagian 3 juga terdapat tombol *addcart* yang berfungsi untuk memasukan produk kekeranjang belanja, bagian nomor 4 merupakan bagian ulasan dari produk tersebut, ulasan terdiri dari gambar profil yang memberi ulasan, nama dan juga pesan ulasan produk tersebut, pada bagian nomor 5 merupakan *detail* dari penjual produk tersebut, yang terdiri dari nama, gambar profil, lokasi dan reputasi penjual, pada bagian 6 merupakan *navigasi* kategori produk dan pada bagian nomor 7 merupakan produk acak yang ditampilkan, sedangkan nomor 8 merupakan bagian dari *footer*,



Gambar 4.18 Peracangan antarmuka halaman *detail* produk

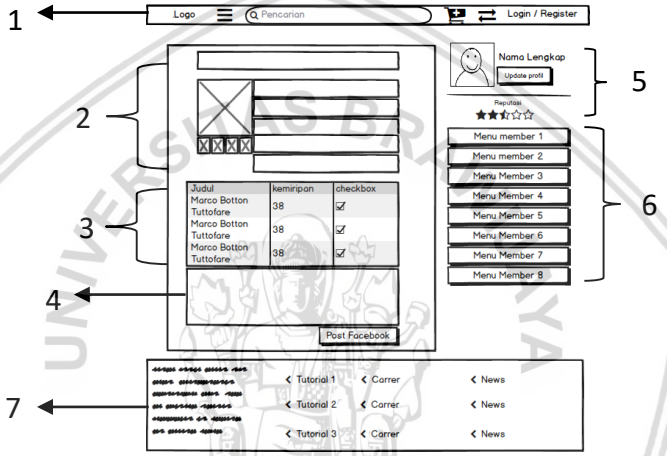
3. Perancangan antarmuka halaman *koneksi facebook*

Pada gambar 4.19 merupakan perancangan antarmuka halaman koneksi *facebook.* Pada gambar tersebut terdapat 7 bagian, bagian nomor 1 yaitu *header*, bagian nomor 2 yaitu tombol yang berfungsi untuk melakukan koneksi dengan facebook, bagian nomor 3 merupakan detail profil facebook yang telah berhasil melakukan konek ke sistem, bagian nomor 5 merupakan detail dari akun yang masuk atau *login* pada saat itu, yang berisi nama lengkap, tombol *update* profil dan reputasi, bagian nomor 6 merupakan menu-menu yang tersedia pada halaman member yang telah melakukan *login* atau masuk ke sistem dan bagian nomor 7 merupakan bagian *footer*



Gambar 4.19 Perancangan antarmuka halaman koneksi facebook

4. Peracangan antarmuka halaman bagikan facebook



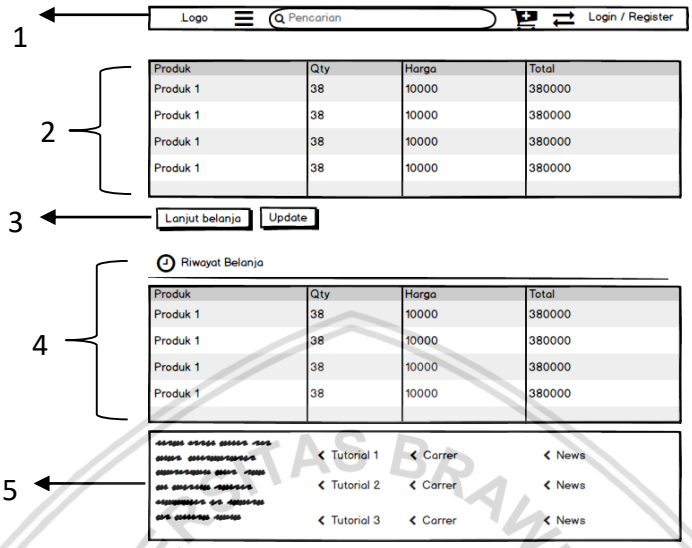
Gambar 4.20 Perancangan antarmuka halaman bagikan *facebok*

Pada gambar 4.20 merupakan gambar perancangan antarmuka halaman cosine similarity ataupun bagikan *facebook*. Pada gambar tersebut dibagi menjadi 7 bagian, bagian nomor 1 merupakan *header.* Bagian nomor 2 merupakan detail dari produk yang dibuka yang meliputi, judul produk, jumlah, kategori, subkategori dan lain sebagainya. Bagian 3 merupakan hasil dari perhitungan kemiripan *grup facebook* dengan produk yang meliputi nama *grup*, nilai kemiripan, dan chekbox. Bagian nomor 4 merupan textarea yang berfungsi untuk menambahkan keterangan saat dilakukan posting ke *grup facebook* pada bagian bawah text area terdapat tombol *post facebook.* Bagian nomor 5 merupakan detail dari member yang *login* atau masuk meliputi nama *member,* tombol *update* profil dan reputasi member tersebut. Bagian nomor 6 merupakan menu member dan bagian nomor 7 merupakan *footer.*

5. Perancangan antarmuka halaman keranjang belanja

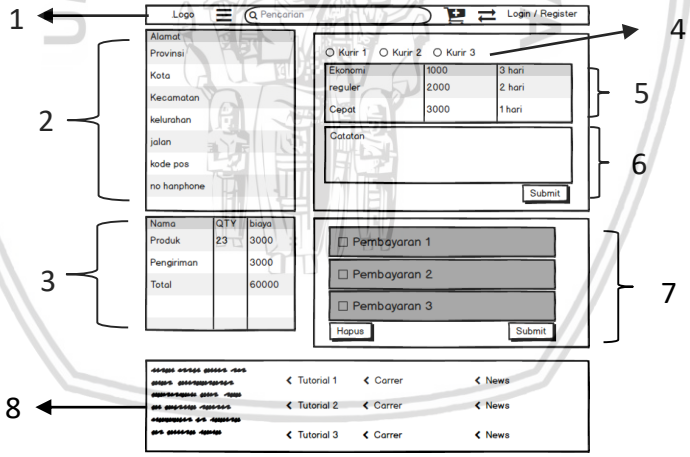
Pada gambar 4.21 merupakan perancangan antarmuka halaman keranjang belanja. Pada gambar tersebut terdapat 5 bagian. Bagian nomor 1 merupakan *header*. Bagian nomor 2 merupakan tabel dari produk yang telah ditambahkan pada keranjang belanja. Bagian nomor 3 merupakan

tombol update keranjang belanja dan juga tombol lanjutkan belanja. Bagain nomor 4 merupakan riwayat belanja yang telah dilakukan dan bagian nomor 5 merupakan bagian *footer*.



Gambar 4.21 Perancangan antarmuka halaman keranjang belanja

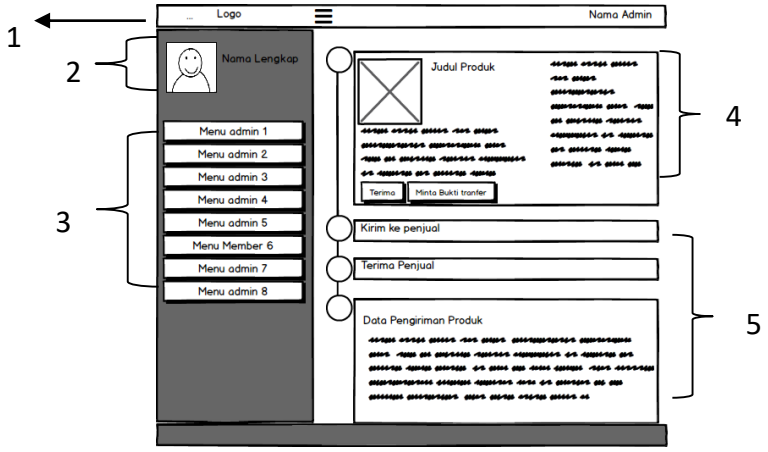
6. Perancangan antarmuka halaman checkout



Gambar 4.22 Perancangan antarmuka halaman checkout

Pada gambar 4.22 merupakan perancangan antarmuka halaman checkout. Pada gambar tersebut dibagi menjadi 8 bagian. Bagian nomor 1 merupakan header. Bagian nomor 2 merupakan alamat tujuan pengiriman barang. Bagian nomor 3 merupakan total jumlah yang harus dibayar. Bagian nomor 4 merupakan kurir yang tersedia dari penjual. Bagian nomor 5 merupakan tabel biaya kurir yang tersedia. Bagian nomor 6 merupakan *text input* untuk catatan pembelian dan pada bawahnya juga tersedia tombol *submit*. Bagian nomor 7 merupakan opsi pembayaran yang tersedia. Dan bagian nomor 8 merupakan *footer*

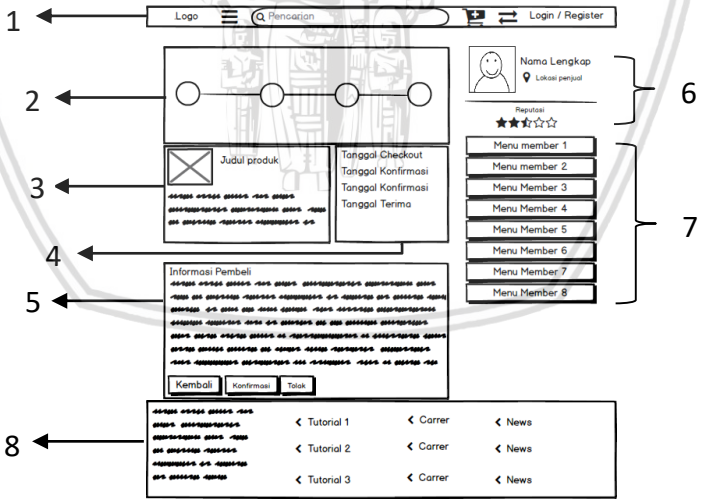
7. Perancangan antarmuka halaman konfirmasi produk admin



Gambar 4.23 Perancangan antarmuka halaman konfirmasi produk admin

Pada gambar 4.23 merupakan perancangan antarmuka halaman konfirmasi produk admin. Pada gambar tersebut dibagi menjadi 5 bagian. Bagian nomor 1 merupakan *header*. Bagian nomor 2 merupakan *detail* dari admin yang sedang *login* atau masuk sistem *detail* tersebut meliputi nama dan foto profil. Bagian nomor 3 merupakan menu admin yang tersedia dibagian *panel admin.* Bagian nomor 4 merupakan detail barang yang dibeli dan juga alamat tujuan, pada bagian tersebut juga terdapat tombol terima dan minta bukti pembayaran. Dan bagian nomor 4 merupakan keterangan dari proses transaksi tersebut.

8. Perancangan antarmuka halaman konfirmasi produk penjual



Gambar 4.24 Perancangan antarmuka konfirmasi produk penjual

Pada gambar 4.24 merupakan perancangan antarmuka konfirmasi produk penjual. Pada gambar ini dibagi menjadi 8 bagian. Bagian nomor 1 merupakan *header.* Bagian nomor 2 merupakan tampilan proses transaksi dikonfirmasi sampai produk dikirim. Bagian nomor 3 merupakan detail produk yang dilakukan transaksi. Bagian nomor 4 merupakan tangal saat produk dilakukan konfirmasi dan sebaginya. Bagian nomor 5 merupakan informasi pembeli yang meliputi nama, alamat dan lain sebagainya yang

digunakan untuk melakukan pengiriman produk. Bagian nomor 6 merupakan *detail* profil yang sedang login atau masuk pada saat itu, yang meliputi nama, foto profil, *update* profil dan reputasi. Bagian nomor 7 merupakan menu yang tersedia *dimember area.* Dan bagian nomor 8 merupakan *footer*

BAB 5 IMPLEMENTASI SISTEM

5.1 Spesifikasi Sistem

Pada bagian ini akan menjelaskan tentang spesifikasi sistem yang digunakan dalam melakukan pembangunan sistem *marketplace* yang dapat merekomendasikan grup facebook yang sesuai produk dengan *algoritme cosine similarity.* Spesifikasi sistem yang digunakan dalam implementasi sistem dibagi menjadi dua yaitu Spesifikasi *Hardware* dan Spesifikasi *Software.*

5.1.1 *Spesifikasi Hardware*

Pada bagian ini akan menjelaskan *Spesifikasi Hardware* yang digunakan dalam mengembangkan sistem *marketpalce,* tabel 5.1 akan menjelaskan tetang *Spesifikasi Hardware* yang digunakan.

Tabel 5.1 *Spesifikasi Hardware*

Hardware	**Keterangan**
Harddisk	500 GB
Processor	AMD A6 6130 APU
RAM	6 GB DDR3L
Graphic Card	AMD Radeo R4 Graphics

5.1.2 *Spesifikasi Software*

Pada bagian ini akan menjelaskan *Spesifikasi Software* yang digunakan dalam mengembangkan sistem *marketplace,* tabel 5.2 akan menjelaskan tentang *Spesifikasi Software* yang digunakan.

Tabel 5.2 *Spesifikasi Software*

Hardware	**Keterangan**
Sistem Operasi	Windows 10 Home 64-bit
Tools	Sublime Text 3, Google Chrome
Bahasa Pemrograman	PHP, HTML, CSS, Javascript
DBMS	MySQL
Server	Localhost (WAMPSERVER)
Framework	Codeigniter 3
Facebook API	Graph API V2.9

5.2 Implementasi Sistem *Marketplace*

Implementasi sistem *marketplace* yang dapat merekomendasikan grup facebook yang sesuai produk dengan algoritme *cosine similarity* dikembangkan dengan perancangan yang telah dilakukan pada bab sebelumnya. Implementasi sistem *marketplace.* Terdiri dari Implementasi Algoritme dan Implementasi Antarmuka.

5.2.1 Implementasi Algoritme

Pada bagian implementasi algoritme menjelaskan mengenai algoritme yang diterapkan pada fungsi-fungsi sistem *marketplace* yang telah dirancang pada pembahasan bab sebelumnya.

A Fungsi *Connect Facebook.*

Pada tabel 5.3 merupakan fungsi dari algoritme *connect facebook*. Pada bagian *connect facebook* ini sistem membutuhkan Graph API yang digunakan untuk menghubungkan antara sistem dan *facebook* pengguna. Pada implementasi ini sistem menggunakan *graph API* versi 2.9. untuk mendapatkan akses *graph API* harus melakukan pendaftaran aplikasi terlebih dahulu, hal tersebut bertujuan untuk mendapatkan *app_id* dan akses *token*, kemudian untuk menerapkan *graph API* pada implementasi ini membutuhkan *library facebook* dan juga *sdk facebook,* setelah *library facebook* dan *sdk facebook* dipasangkan pada sistem untuk memanggil fungsi *graph API* yang dinginkan tinggal melakukan pemanggilan *library* diikuti dengan fungsi yang diinginkan seperti facebook->is_authenticated() yang berfungsi untuk *autentifikasi facebook* pada sistem, seperti pada implementasi algoritme *connect facebook* ditabel 5.3. Pada fungsi *conect facebook* terdapat dua kondisi pertama jika kondisi member belum pernah melakukan konek ke facebook dengan sistem *marketplace* ini maka data yang member akan ditambahkan terlebih dahulu ke *database* , data yang ditambah kedatabase meliputi *oauth_uid,firstname, lastname, email, gender*, locale, dan alamat profile facebook member. Kondisi kedua jika member tersebut sudah pernah *connect facebook* dengan sistem ini maka user tersebut langsung login tanpa perlu sistem menambahkan data *facebook* ke *database.*

Tabel 5.3 Implementasi algoritme *connect facebook*

```
1  public function koneksifb()
2        {
3               $data = array("title"      => "member");
4               $userData = array();
5               if($this->facebook->is_authenticated()){
6                     $userProfile = $this->facebook->request('get',
7  '/me?fields=id,first_name,last_name,email,gender,locale,picture.wid
8  th(800).height(800)');
9
10             // masukkan ke database
11             $userData['member_id'] = $this->session-
12 >userdata('member_id');
13             $userData['oauth_provider'] = 'facebook';
```

```
            $userData['oauth_uid'] = $userProfile['id'];
            $userData['first_name'] = $userProfile['first_name'];
            $userData['last_name'] = $userProfile['last_name'];
            $userData['email'] = $userProfile['email'];
            $userData['gender'] = $userProfile['gender'];
            $userData['locale'] = $userProfile['locale'];
            $userData['profile_url'] =
'https://www.facebook.com/'.$userProfile['id'];
            $userData['picture_url'] =
$userProfile['picture']['data']['url'];

            // Insert or update user data
            $userID = $this->member->checkUser($userData);

                    // Check user data insert or update status
            if(!empty($userID)){
                $data['userData'] = $userData;
                $this->session->set_userdata('userData',$userData);
            } else {
               $data['userData'] = array();
            }

                    // Get logout URL
                    $data['logoutUrl'] = $this->facebook-
>logout_url();
             }else{
            $fbuser = '';

                    // Get login URL
            $data['authUrl'] =  $this->facebook->login_url();
        }

             // Load login & profile view

             $data['member'] = $this->member-
>cek_login(array('member_id'=>$this->session-
>userdata('member_id')));
            $this->front->views('front/koneksiFacebook',$data);
       }
```

B Fungsi Cosine Similarity

Tabel 5.4 Implementasi fungsi *cosine similarity*

```
//potongan dari class Cosinesimilarity
static public function tags_to_point($articles) {
             $tags = array();
             foreach($articles as $article) {
                    $tags = array_merge($tags, $article['tags']);
             }
             $tags = array_unique($tags);

             $tags = array_fill_keys($tags, 0);
             ksort($tags);
             return $tags;
       }

```

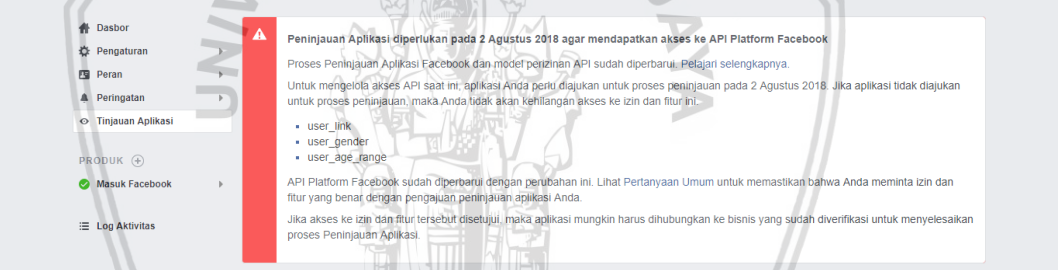
```
14          static public function cosine($a, $b) {
15                  ksort($a);
16                  ksort($b);
17                  //dot_product
18                  $products = array_map(function($a, $b) {
19                          return $a * $b;
20                  }, $a, $b);
21                  $dot_products=array_reduce($products, function($a, $b)
22  {
23                          return $a + $b;
24                  });
25                  //magnitude a
26                  $squares = array_map(function($x) {
27                          return pow($x, 0.5);
28                  }, $a);
29                  $magnitude_a=sqrt(array_reduce($squares, function($a,
30  $b) {
31                          return $a + $b;
32                  }));
33                  //magnitude b
34                  $squares = array_map(function($x) {
35                          return pow($x, 0.5);
36                  }, $b);
37                  $magnitude_b=sqrt(array_reduce($squares, function($a,
38  $b) {
39                          return $a + $b;
40                  }));
41
42                  return $dot_products/($magnitude_a * $magnitude_b) *
43  100;
44
45          }
46          public function perhitungan()
47          {
48                  $data = array("title"       => "member");
49                  $produk_id = $this->uri->segment(3);
50                  $produk = $this->produk->getProdukByid($produk_id)-
51  >result_array();
52                  $idkategori= $produk[0]['kategori_id'];
53                  $kategori = $this->kategori-
54  >getKategoriById($idkategori)->result_array();
55                  $idsubkategori= $produk[0]['subkategori_id'];
56                  $subkategori = $this->kategori-
57  >getSubKategoriById($idsubkategori)->result_array();
58                  $kata_produk =
59  $produk[0]['judul_produk']."".$produk[0]['deskripsi']."
60  ".$kategori[0]['judul_kategori']."
61  ".isset($subkategori[0]['judul_subkategori']);
62                  $pisah_produk= explode(" ",$kata_produk);
63
64                  $where = array(
65                          'member_id'   => $this->session-
66  >userdata('member_id')
67                          );
68                  $grup= $this->grup->getListGrupById($where)-
69  >result_array();
70                  $x=0;
                    foreach ($grup as $key) {
```

```
                        $grupfb = $this->facebook->request('get',
'/'.$key['id_grup'].'?fields=name,description,feed.limit(3)');
                        if (!empty($grupfb['feed']['data'])) {
                               foreach ($grupfb['feed']['data'] as $text)
{
                                      $textjual = $text['message'];
                               }
                        }
                        $kata_grup =!empty($grupfb['name'])."
".!empty($grupfb['description'])."
".!empty($textjual)."".$key['judul'];
                        $pisah_grup[$x]= explode(" ",$kata_grup);
                        $x++;
                }
                for ($i=0; $i < $x; $i++) {
                        $grupfb = $this->facebook->request('get',
'/'.$grup[$i]['id_grup'].'?fields=name');
                        $articles[$i]['id_grup']=$grup[$i]['id_grup'];

        $articles[$i]['nama_grup']=!empty($grupfb['name']);
                        $articles[$i]['tags']=$pisah_grup[$i];
                }


                $target = $pisah_produk;
                $tags = $this->tags_to_point($articles);
                $compare = array_fill_keys($target, 1) + $tags;
                $x=0;
          foreach($articles as $article) {
              $ak = array_fill_keys($article['tags'], 1) + $tags;
              $cs[$x]['id_grup']=$article['id_grup'];
              $cs[$x]['nama_judul']=$article['nama_grup'];
              $cs[$x]['score']=number_format($this->cosine($compare,
$ak),2);
              $x++;
          }
                $data ['cs']        = $cs;
                $data ['member']    = $this->member-
>cek_login(array('member_id'=>$this->session-
>userdata('member_id')));
                $data ['produk']    = $this->produk-
>getProdukByid($produk_id)->row_array();

                $this->front->views('front/produk/cosine', $data);
        }
```

Pada tabel 5.7 merupakan implementasi *algoritme cosine similarity*. Pada fungsi ini terdapat beberapa data yang diambil dari data produk dan juga kategori untuk dilihat tingkat kemiripan terhadap grup facebook yang tersedia oleh member. Untuk mengambil data dari produk yang akan diproses maka perlu id dari produk tersebut, sehinga untuk memanggi id dari produk tersebut maka digunakan fungsi *uri segment* pada *framework codeigniter.* Setelah dapat id dari produk tersebut, maka akan ditampilkan kembali pada dengan penggunakan parameter produk tersebut. Begitu juga dengan ketegori yang kemudian dipanggil dan di masukan pada parameter $kata_produk, parameter $kata_produk tersebut digunakan sebagian kata utama yang dicocokan dengan *grup facebook*. Untuk mengambil data dari grup facebook diperlukan koneksi ke

facebook terlebih dahulu, hal tersebut dikarenakan *graph API facebook* hanya dapat berfungsi jika user sudah melakukan *autentifikasi facebook* atau *login facebook* terlebih dahulu. Jika member sudah melakukan koneksi ke *facebook* maka sistem dapat melakukan pemanggilan data grup facebook sesuai dengan grup yang telah ditambahkan oleh member, data grup facebook yang diambil dengan menggunakan graph API meliputi nama grup, deskripsi grup dan juga postingan grup. Semua data grup facebook yang terdapat pada member tersebut akan dicocokan dengan produk dan kategori produk yang telah dimasukan pada parameter $kata_produk. Kemudian akan dilakukan perhitungan dan mengahasilkan nilai tingkat kemiripannya pada setiap grup.

Pada saat implementasi *cosine similarity* pada pembangunan sistem *marketplace* ini penulis menggunakan Graph API versi 2.9 dan pada saat dokumen ini tulis graph API versi 2.9 telah diberhentikan aksesnya dikaranakan pada *facebook* mengalamai masalah. Kemudian digantikan graph API versi 3, pada *graph API* versi 3 aplikasi harus melaui proses peninjauan ulang untuk mendapatkan persetujuan oleh pihak *facebook*, sehingga aplikasi yang belum mendapat persetujuan dari *facebook* tidak akan bisa berjalan seperti pada aplikasi sebelumnya. Dan pada *graph API* versi 3 ini peninjauan baru akan dimulai pada tanggal 2 agustus 2018. Seperti pada gambar berikut



Gambar 5.1 Informasi peninjauan graph API versi 3(**Sumber**: https://developers.facebook.com)

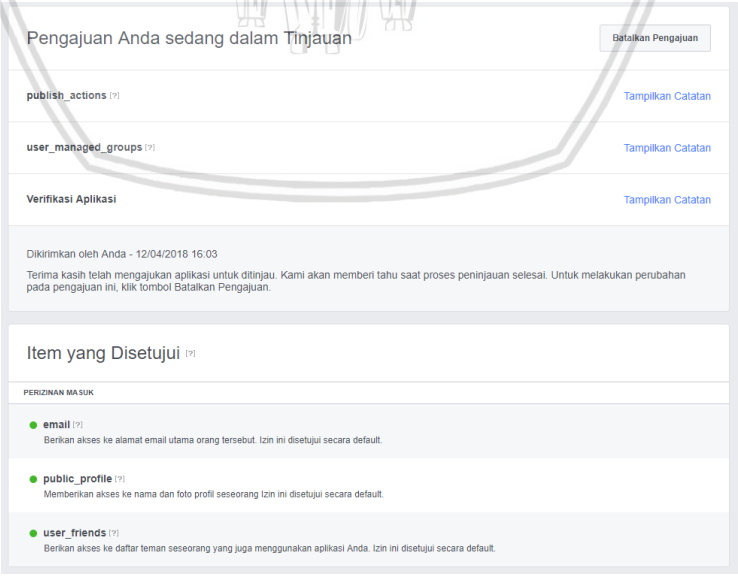
C Fungsi *Post Facebook*

Tabel 5.5 Implementasi Algoritme *post facebook*

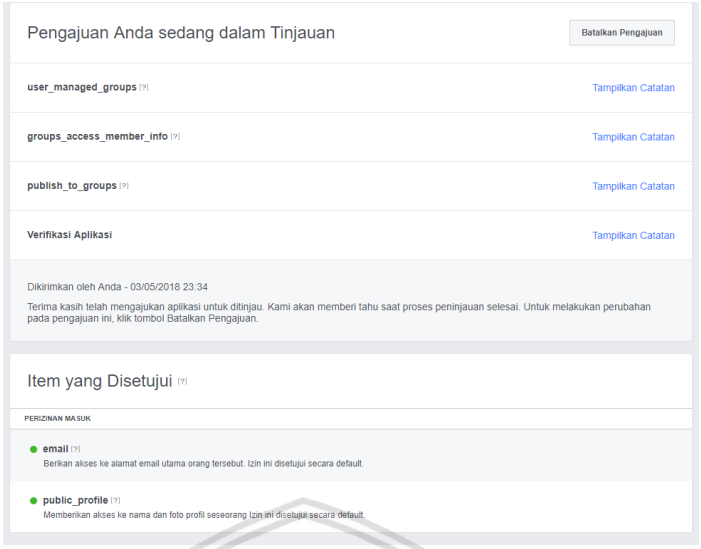
```
1  //potongan dari class Cosinesimilarity
2  public function postproduk(){
3              $checkbox = $this->input->post('mycheckbox');
4              $x=0;
5              foreach ($checkbox as $idg) {
6                     $gdetail[$x] = $this->grup->getGrupById($idg)-
7  >row_array();
8                     $data[$x] = $this->facebook-
9  >request('post','/'.$idg.'/feed', array(
10                                   'link' => base_url(),
11                                   'message' =>
12 $gdetail[$x]['tag'].$this->input-
13 >post('deskripsi').strip_tags($this->input->post('desc')),
14                    ));
15                    $x++;
               }
```

16	}

Pada tabel 5.5 merupakan implementasi dari algoritme *post facebook.* Pada algoritme *post facebook* ini berfungsi untuk melakukan *post* pada *grup facebook* yang telah dipilih oleh pengguna. Terdapat variabel checkbox yang berfungsi mengambil nilai input dari nama mycheckbox, pada variabel checkbox akan dilkukan perluangan menggunakan fungsi *foreach,* pada perulangan tersebut terdapat beberapa parameter yang digunakan untuk menadapatkan data dari grup maupun produk tersebut. Pada variabel gdetail berfungsi untuk mendapatkan detail dari grup facebook dengan menggunakan fungsi getGrupByid yang ada pada model *grup*, kemudian akan ditampilkan dalam bentuk *array*. Pada variabel berfungsi untuk melakukan *post* ke *facebook* dengan menggunkan *function request* yang ada pada *libary* facebook ,kemudian melakukukan post menggunkan parameter *link* yang befungsi untuk memberikan *url* produk , pada parameter *message* berfungsi untuk menampilkan pesan atau deskripsi pada *post facebook,* parameter *mesagge* akan berisi berupa *tag* dari detail grup, deskripsi produk dari hasil input yang berisi tentang deskripsi produk dan juga desc yang berisi kalimat tambahan yang ingin disematkan pada postingan *facebook.* Pada variabel gdetail dan data akan dilakukan perulangan sesuai dengan jumlah grup yang diiginkan pengguna. Pada saat implementasi post facebook pada sistem marketplece ini penulis menggunakan *graph API* versi 2.9, pada graph API versi 2.9 untuk melakukan post ke facebook atau posting ke grup facebook membutuhkan akses API *public_action* dan *user_manage_grup* agar dapat melakukan posting ke grup facebook seperti pada gambar 5.2. Dan pada saat dokumen ini ditulis graph API versi 2.9 sudah diberhentikan dan digantikan ke versi 3. Pada graph API versi 3 ini akses API *public_action* sudah dihilangkan dan digantikan dengan *publish_to_grups* seperti pada gambar 5.3.



Gambar 5.2 Akses Graph API versi 2.9 (Sumber: https://developers.facebook.com)



Gambar 5.3 Akses Graph API versi 3 (Sumber: https://developers.facebook.com)

Untuk menggunakan *graph API* versi 3 harus melakukan peninjauan ulang terlebih dahulu, dan peninjauan pada app yang menggunakan *graph API* versi 3 baru akan dimulai pada 2 Agustus 2018.

D Fungsi tambah Item keranjang belanja

Pada tabel 5.6 merupakan implementasi fungsi tambah item pada keranjang belanja. Pada tedapat dua kondisi, dimana jika kondisi form input telah di submit maka nilai dari form input tersebut akan diteruskan ke fungsi addcart pada model transaksi untuk dilakukan penulisan pada database. Jika proses sudah selesai maka tetap akan berada pada halaman produk tersebut, karena fungsi *redirect* tetap diarahkan sesuai input slug yang tersedia.

Tabel 5.6 Implementasi Tambah Item Keranjang Belanja

```
1  //potongan dari class keranjangBelanja
2  public function addcart(){
3                 $slug =$this->input->post('slug');
4                 if(isset($_POST['submit'])){
5                         $this->transaksi->addcart();
6                         redirect('produk/detail/'.$slug);
7                 }
8                 else {
9                         redirect('produk/detail/'.$slug);
10                }
11        }
```

E Fungsi *Checkout*

Pada tabel 5.7 merupakan implementasi algoritme dari fungsi *checkout*. Pada fungsi checkout ini terdapat dua kondisi, kondisi pertama jika form input telah terisi dan submit maka data dari *form input* tersebut akan diteruskan dengan fungsi *updateItem* pada model transaksi, kemuidan akan dilakukan update sesuai dengan produk yang dibeli. Pada kondisi kedua apabila form input tidak disubmit maka akan menampilkan detail dari transaksi tersebut, dari detail pembeli,

alamat pembeli, produk, penjual, pengiriman yang tersedia dan alamat pengiriman.

Tabel 5.7 Implementasi Fungsi Checkout

```
//potongan dari class keranjangBelanja
public function checkout(){
            if(isset($_POST['submit'])){
                  $this->transaksi->updateItem();
                  redirect('KeranjangBelanja/checkout/'.$this->input->post('transaksi_id'));
            }else{
                  $transaksi_id = $this->uri->segment(3);
                  $data['transaksi'] = $this->transaksi->getTransaksiById($transaksi_id)->row_array();
                  //data pembeli
                  $member_id = $this->session->userdata('member_id');
                $data['pembeli']            = $this->member->getMemberById($member_id)->row_array();
getListAlamatById($member_id)->result();
                  // detail alamat tujuan
      $alamat_buyer                 = $data['transaksi']['almt_penerima'];
                  $data['almt_tujuan']      = $this->alamat->getAlamatByIdBuyer($alamat_buyer)->row_array();
                //data produk yang dibeli
                  $produk_id                              = $data['transaksi']['produk_id'];
                  $data['produk']                 = $this->produk->getProdukByid($produk_id)->row_array();
                  //data seller
                  $member_id                              = $data['produk']['member_id'];
                $data['seller']             = $this->member->getMemberById($member_id)->row_array();

                // kurir
                $data['pengiriman']         = $this->member->getPengiriman($member_id)->result_array();

                $alamat_seller = $data['seller']['alamat_id'];
                $data['alamat_sell']   = $this->alamat->getAlamatByIdSeller($alamat_seller)->row_array();

                $this->front->views('front/transaksi/pagecheckout',$data);
            }
      }
```

F Fungsi konfirmasi transaksi produk Admin

Pada tabel 5.8 merupakan implementasi algoritme dari konfirmasi admin. Pada fungsi konfirmasi admin ini di bagi dalama 3 *case*, *case* pertama berfungsi untuk menerima order dari member, pada case pertama ini status data transaksi akan

diubah menjadi 1 dan dimasukanya waktu penerimaan order kemudian dua data tersebut akan d*iupdate* pada data transaksi menggunakan fungsi updateTransaksi pada fungsi model transaksi. Pada *case* kedua berfungsi untuk melakukan pembatalan penerimaan order, diaman variabel param yang berfungsi untuk merubah status transaksi yang bernilai 1 menjadi 0 yang kemudian akan dilakukan perubahan dengan fungsi updateTransaksi dengan menggunakan fungsi pada model transaksi. *Case* ketiga berfungsi untuk mengubah nilai pada data gambar menjadi 88, pada data gambar berisi nilai 88 maka sistem secara otomatis akan meminta bukti tranfer ke pembeli dengan cara mengunggah foto bukti tranfer ke sistem, dengan demikian admin akan lebih mudah untuk melakukan verifikasi.

Tabel 5.8 Implementasi Algoritme konfirmasi transaksi produk Admin

```
1  //potongan dari class DataTransaksi
2  public function Update($id, $action = null){
3
4              $updated = false;
5               switch ($action) {
6              case 1:                    //approve status
7              $updated = true;
8                   $param = array('status' => '1',
9               'time_approveadmin' =>date('Y:m:d h:m:s'));
10             $where = "invoice = '".$id."'";
11             $this->transaksi->updateTransaksi($param, $where);
12             $this->getDetailTransaksi($id);
13             break;
14             case 2:                      // de-approve status
15             $updated = true;
16             $param = array('status' => '0');
17             $where = "invoice = '".$id."'";
18             $this->transaksi->updateTransaksi($param, $where);
19             $this->getDetailTransaksi($id);
20                     break;
21                     case 3:
22                     $updated = true;
23             $param = array('gambar' => '88');
24             $where = "invoice = '".$id."'";
25                     $this->transaksi->updateTransaksi($param,
26 $where);
27                     $this->getDetailTransaksi($id);
28                     break;
29         }
30
31         if ($updated == false) {
32             $this->template->views('admin/dataMember/',$data);
33         }
       }
```

G Fungsi konfirmasi transaksi produk penjual

Pada tabel 5.9 merupakan implementasi algoritme konfirmasi penjual. Pada implementasi algoritme konfirmasi penjual ini terdapat 3 kondisi, kondisi pertama berfungsi untuk menerima transaksi baru yang akan diproses. Jika kondisi pertama dijalankan maka variable param akan berisi *array* status yang

berisi nilai 2 dan *time_confirmseller* yang berisi waktu pada saat kondisi pertama ini dijalankan. Kemudian dari data yang terdapat pada variabel param tersebut akan diupdate bedasarkan variabel *where* yang berisi data *invoice*, kemudian akan dilakukan update pda model transaksi yang bedasarkan variabel param dan where. Setelah update berhasil akan dialihkan kehalam detail melaui function getDetail yang bedasarkan id atau *invoice* transaksi tersebut. Kondisi kedua berfungsi untuk membatalkan konfirmasi yang telah diterima. Jika kondisi kedua dijalankan maka *vaiabel* param yang berisi array status dengan nilai 1, data status dari variable param tersebut akan digunakan untuk melakukan update pada transaksi yang sudah diterima dengan nilai status 2. Kemudian pada variable *where* yang berisi data invoice transaksi yang ingin diupdate, kemudian akan dilakukan *update* pada model transaksi pada *function* updateTransaksi, setelah melakukan update berhasil maka akan dialikan ke halaman detail dengan menggunakan *funtion* getDetail yang bedasarkan *invoice* transaksi.pada kondisi ketiga merupakan kondisi yang digunakan untuk menolak transaksi baru, pada kendisi ketiga ini akan mengubah nilai pada status menjadi 5. Pada *variabel* param yang berfungsi untuk mengubah status menjadi 5. Kemudian *variabel where* digunakan untuk mencocokan transaksi mana yang akan dirubah, dari variabel param dan where tesebut kemudian akan dilempar ke model transaksi *function* updateTransaksi, dan kemudian akan dialihkan ke halaman detail transaksi tersebut.

Tabel 5.9 Implementasi Algoritme konfirmasi penjual

```
1  //potongan dari class Transaksi
2  public function Update($id, $action = null){
3              $updated = false;
4                switch ($action) {
5              case 1:                         //approve status
6              $updated = true;
7              $param = array('status' => '2',
8                                        'time_confirmseller' =>date('Y:m:d
9  h:m:s'));
10             $where = "invoice = '".$id."'";
11             $this->transaksi->updateTransaksi($param, $where);
12             $this->session->set_flashdata('msg', 'status1');
13             $this->getDetail($id);
14             break;
15             case 2:                         // de-approve status
16             $updated = true;
17             $param = array('status' => '1');
18             $where = "invoice = '".$id."'";
19             $this->transaksi->updateTransaksi($param, $where);
20             $this->session->set_flashdata('msg', 'status99');
21             $this->getDetail($id);
22             break;
23             case 3:                         // tidak menerima
24             $updated = true;
25             $param = array('status' => '5');
26             $where = "invoice = '".$id."'";
27             $this->transaksi->updateTransaksi($param, $where);
28             $this->session->set_flashdata('msg', 'tolak');
               $this->getDetail($id);
```

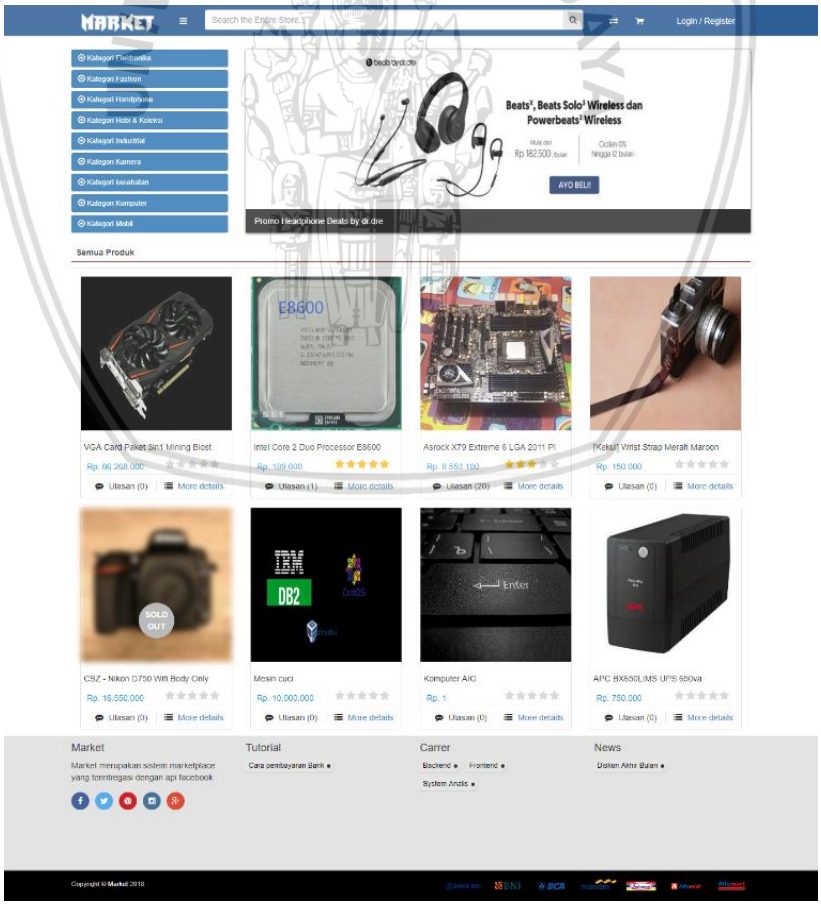
```
29              break;
30          }
31          if ($updated == false) {
32              redirect('transaksi/getList');
33          }
34      }
35
```

5.2.2 Implementasi Antarmuka

Pada bagian ini merupakan penjelasan dari implementasi antar muka yang telah rancang pada bab sebelumnya. Antarmuka merupakan jembatan interaksi antara pengguna dengan sistem.

A Halaman Utama

Pada gambar 5.4 merupakan halaman utama. Pada halaman utama tersebut sistem akan menampilkan beberapa bagian yaitu terdapat bagian slider gambar kemudian sebelahnya terdapat daftar kategori. Pada bagian bawahnya terdapat 8 produk yang ditampilkan pada halaman utama tersebut. Pada produk tersebut menampilkan gambar produk, nama produk, harga, jumlah ulasan dan reputasi dari produk tersebut.



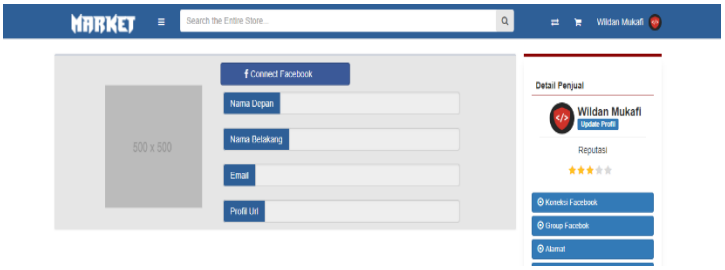
Gambar 5.4 Halaman Utama

B Halaman detail produk dan tambah item keranjang belanja

Pada gambar 5.5 merupakan antarmuka halaman detail produk dan tambah item keranjang belanja. Pada halaman tersebut sistem akan menampilkan *detail* dari produk yang sudah dipilih , selain menampilkan detail produk pada halaman tersebut juga menyediakan tombol *addcart* atau tombol tambah item keranjang belanja, tombol tersebut berfungsi jika user atau pengguna ingin memasukan produk ke keranjang belanja untuk dilakukan pembelian.

Gambar 5.5 Halaman detail produk dan tambah item keranjang belanja

C Halaman Koneksi *facebook*

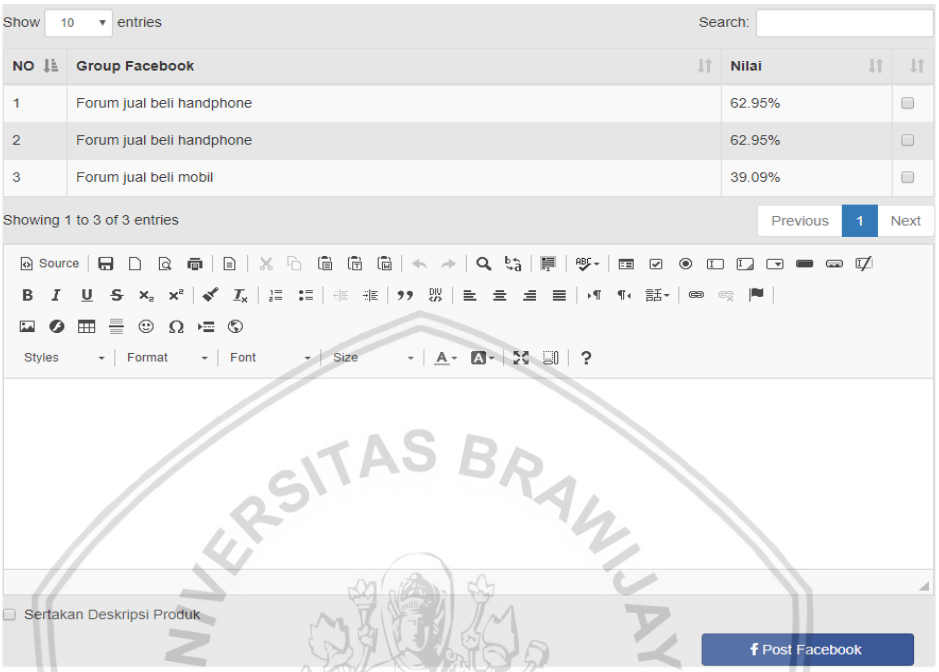


Gambar 5.6 Halaman Koneksi *facebook*

Pada gambar 5.6 merupakan halaman koneksi *facebook*. Pada halaman tersebut berfungsi untuk melakukan koneksi *facebook* dengan sistem

marketplace. Pada halaman tersebut terdapat tombol koneksi *facebook* dan juga foto besertas *text input* yang berfungsi menampilkan data *facebook* yang telah terkoneksi.

D Halaman bagikan grup facebook

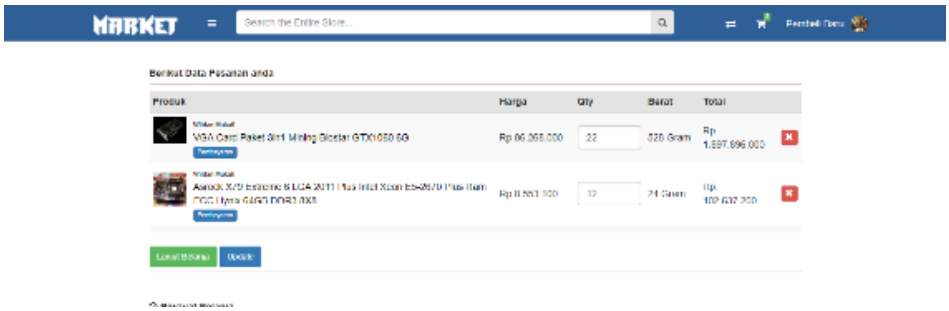


Gambar 5.7 Halaman bagikan grup facebook

Pada halaman 5.7 merupakan antarmuka halaman bagikan *grup facebook*. Halaman ini akan menampilkan nilai kemiripan antara produk dengan grup facebook yang telah dimasukan oleh *member*.

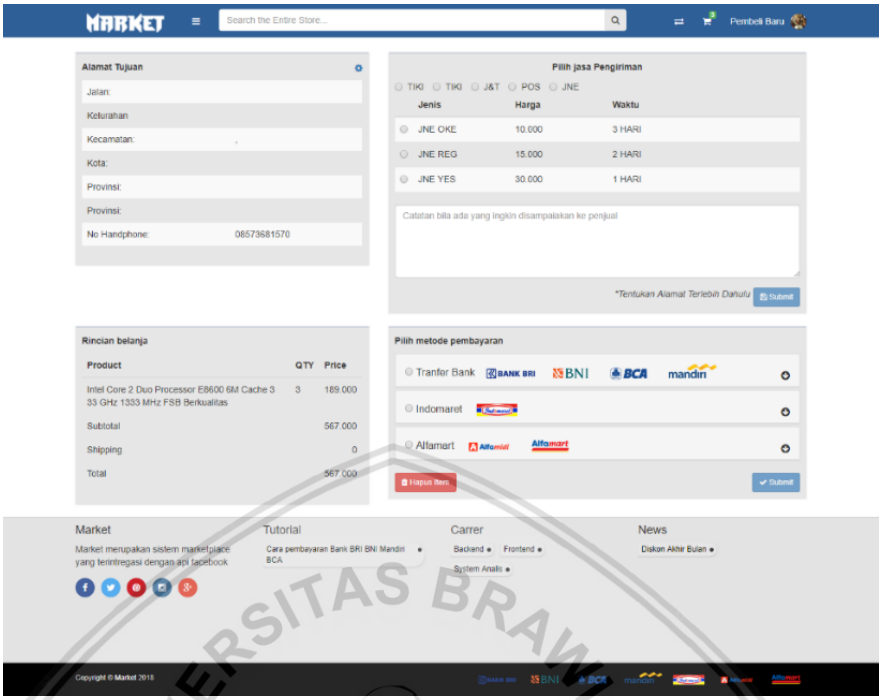
E Halaman keranjang belanja

Pada gambar 5.8 merupakan gambar keranjang belanja dan riwayat belanja. Pada halaman tersebut sistem akan menampilkan daftar produk yang telah dimasukan dalaman keranjang belanja, dan juga pada sistem tersebut sistem juga akan menampilkan daftar belanja yang telah dilakukan pada waktu sebelumnya.



Gambar 5.8 Halaman keranjang belanja

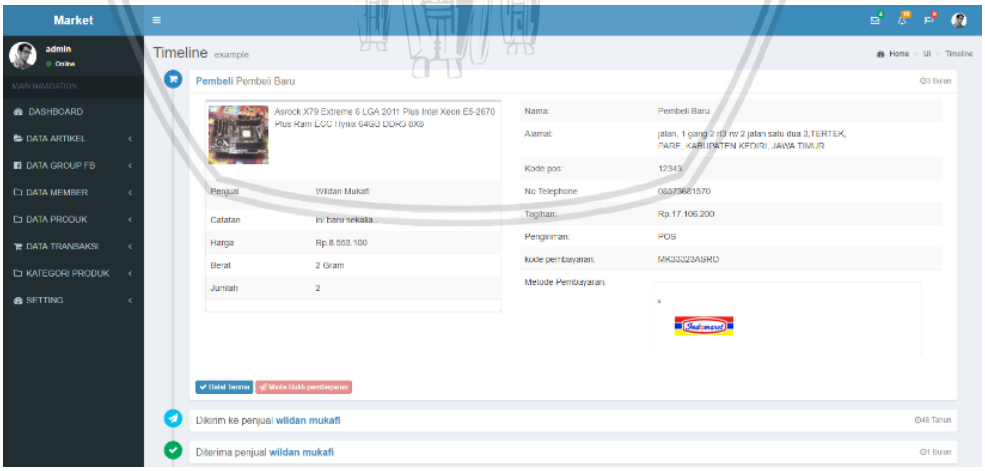
F Halaman *checkout*



Gambar 5.9 Halaman *checkout*

Pada gambar 5.9 merupakan halaman *checkout*. Pada halaman ini berfungsi untuk menlanjutkan proses transaksi pembelian produk, dimana pada proses ini akan menambahan alamat tujuan pengiriman, kuriri yang dipakai, catatan pembelian dan metode pembayaran yang dipakai.

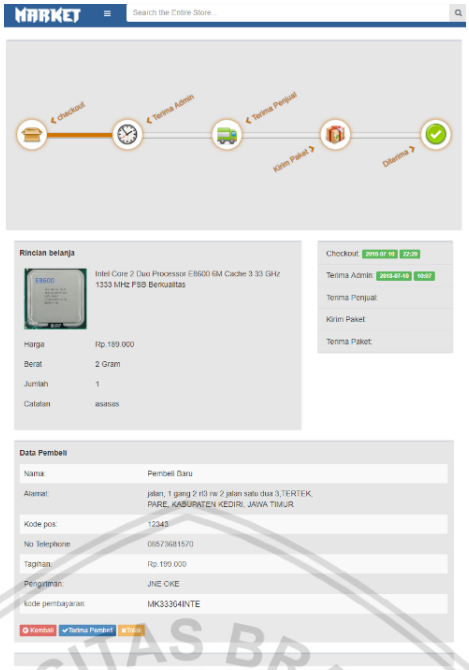
G Halaman Konfirmasi transaksi produk Admin



Gambar 5.10 Halaman Konfirmasi transaksi produk Admin

Pada gambar 5.10 merupakan halaman konfirmasi admin. Halaman ini berfungsi untuk melakukan konfirmasi transaksi yang baru masuk ke sistem yang kemudian akan diteruskan ke penjual untuk diproses pengiriman paket.

H Halaman Konfirmasi transaksi produk Penjual

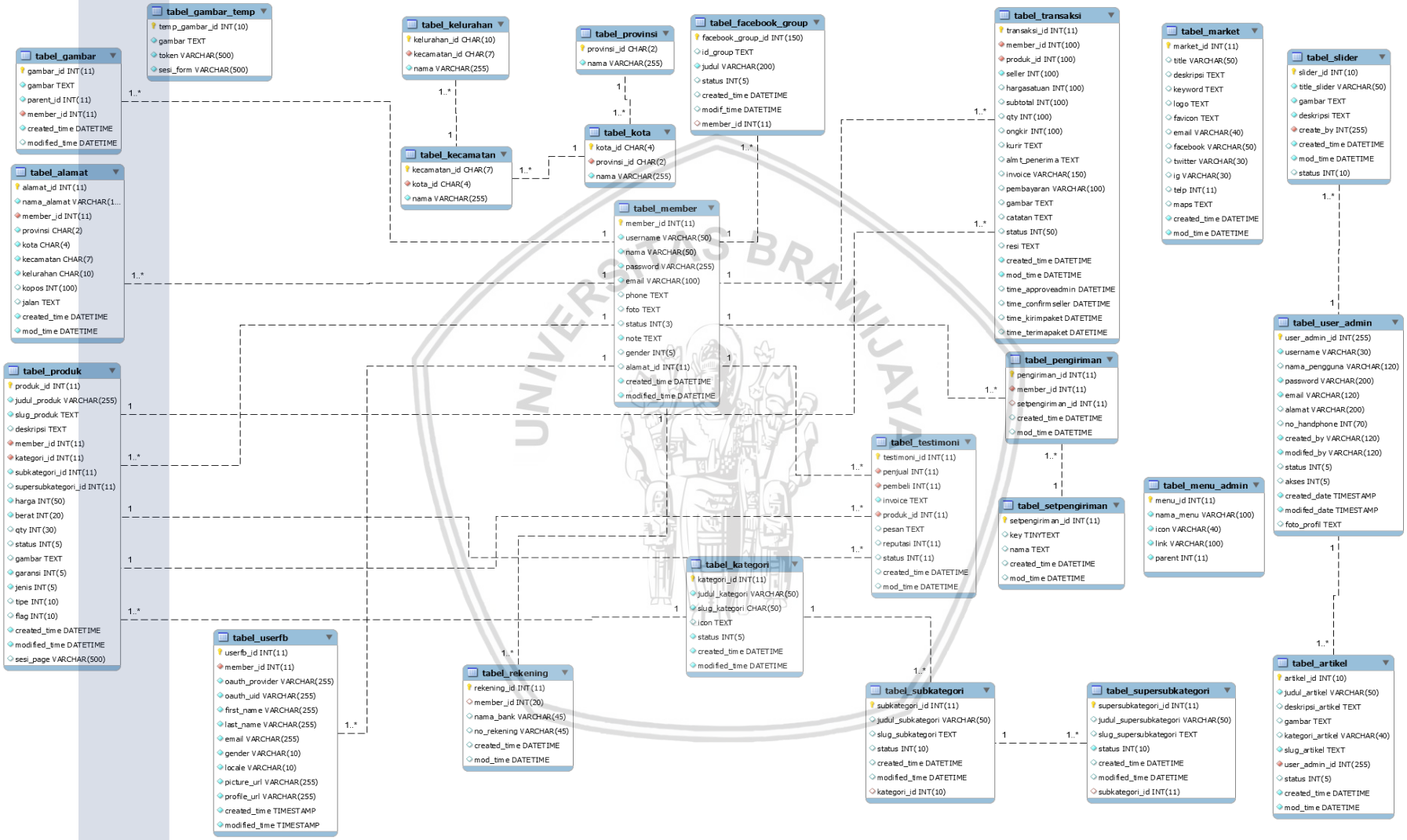


Gambar 5.11 Halaman Konfirmasi transaksi produk Penjual

Pada gambar 5.11 merupakan halaman konfirmasi penjual. Halaman ini berfungsi untuk melakukan konfirmasi transaksi yang masuk setelah melalui konfirmasi admin.

5.2.3 Implementasi Basis data

Pada gambar 5.12 menejaskan tetang implementasi basis data yang digunakan dalam sistem *marketplace*. Dalam implentasi sistem *marketplace* menggunakan 24 tabel *database*. Setiap tabel memiliki kardinalitas antara tabel satu dengan tabel lainya. Tabel produk dengan tabel gambar dan tabel user admin dengan tabel menu admin memiliki kardinalitas *many to one*. Tabel produk dengan tabel member , tabel member dengan tabel userfb dan tabel user admin dengan tabel kategori memiliki kardinalitas *many to many*. Tabel alamat dengan tabel lokasi dan tabel kategori dengan tabel produk memiliki kardinalitas *one to one*, begitupula tabel lainya.



Gambar 5.12 Implementasi Basis Data

BAB 6 PENGUJIAN

Pada bab ini akan dilakukan tahap pengujian dari sistem marketplace yang dapat merekomendasikan *grup facebook* yang sesuai dengan produk menggunakan algoritme cosine similarity. Pengujian dilakukan melalui 3(tiga) tahap yaitu pengujian *white-box*, pengujian *black-box* dan pengujian kebutuhan. Dari hasil pengujian tersebut akan disertakan analisis yang bertujuan untuk memastikan sistem telah memenuhi kebutuhan yang didefinisikan pada tahap sebelumnya.

6.1 Pengujian *White-box*

Pengujian *white-box* merupakan tahap pengujian yang bertujuan untuk memastikan algoritme telah diimplementasikan sesuai dengan yang diharapkan, dengan jenis pengujian *basis-path.* Algoritme yang diuji meliputi koneksi facebook, *addcart*, *checkout*, konfirmasi transaksi produk admin, konfirmasi transaksi produk penjual, fungsi *cosine similarity*, *post facebook*. Dari beberapa operasi tersebut dipilih karena operasi tersebut merupakan domain permasalahan dari sistem yang dibangun.

6.1.1 Pengujian unit

Pengujian unit dalam penelitian ini menggunakan teknik *basis path testing* untuk menguji setiap operasi dari setiap klas berdasarkan dengan algoritme yang dihasilkan pada tahap perancangan. *Algoritme* yang diuji antara lain koneksifb, *Cosinesimilarity, post, addcart, checkout, getInvoice,* transaksi admin, transaksi penjual.

6.1.1.1 Pengujian unit koneksifb

- Nama Klas (controller) : koneksifb.php
- Nama Operasi : index ()
- Proses Uji : Tabel 6.1

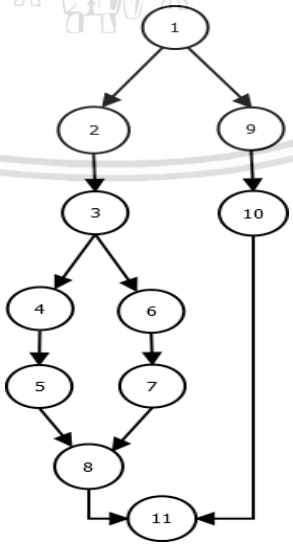
Tabel 6.1 Pengujian Unit koneksi *Facebook*

```
appID = ambil data app facebook developer                          1
appSecret = ambil data app secret facebook developer
accesToken = appId appSecret
IF(facebook = is_authenticated) THEN                               2
        ambil data first_name, last_name, email, gender, picture
        data masukan member_id  = set data session member login
        data masukan first_name = facebook first_name              3
        data masukan last_name  = facebook last_name
```

```
data masukan email     = facebook email
data masukan gender    = facebook gender
data masukan picture   = facebook picture url
datauser = simpan data member_id, first_name, last_name, email,
gender, picture
IF(dataUser !=null) THEN                                        4
    ambil data masukan member_id, first_name,
    last_name, email, gender, picture                           5
    set session member_id, first_name,
    last_name, email, gender, picture
ELSE THEN                                                       6
    menampilkan halaman koneksifb
    menampilkan tombol logout                                   7
END IF                                                          8
ELSE THEN                                                       9
    menampilkan tombol koneksi facebook                         10
END IF                                                          11
```

Basis Path Testing

1. *Flow Graph*



Gambar 6.1 *Flow Graph Operasi index()*

2. *Cyclomatic Complexity*
   - V(G) = Jumlah Region = 3
   - V(G) = E – N + 2 = 12– 11 + 2 = 3
   - V(G) = P + 1 = 2+ 1 = 3
3. *Independent Path*
   - Jalur 1: 1– 2 – 3 – 4 – 5 – 8 – 11
   - Jalur 2: 1– 2 – 3 – 6 – 7 – 8 – 11
   - Jalur 3: 1 – 9 – 10 – 11

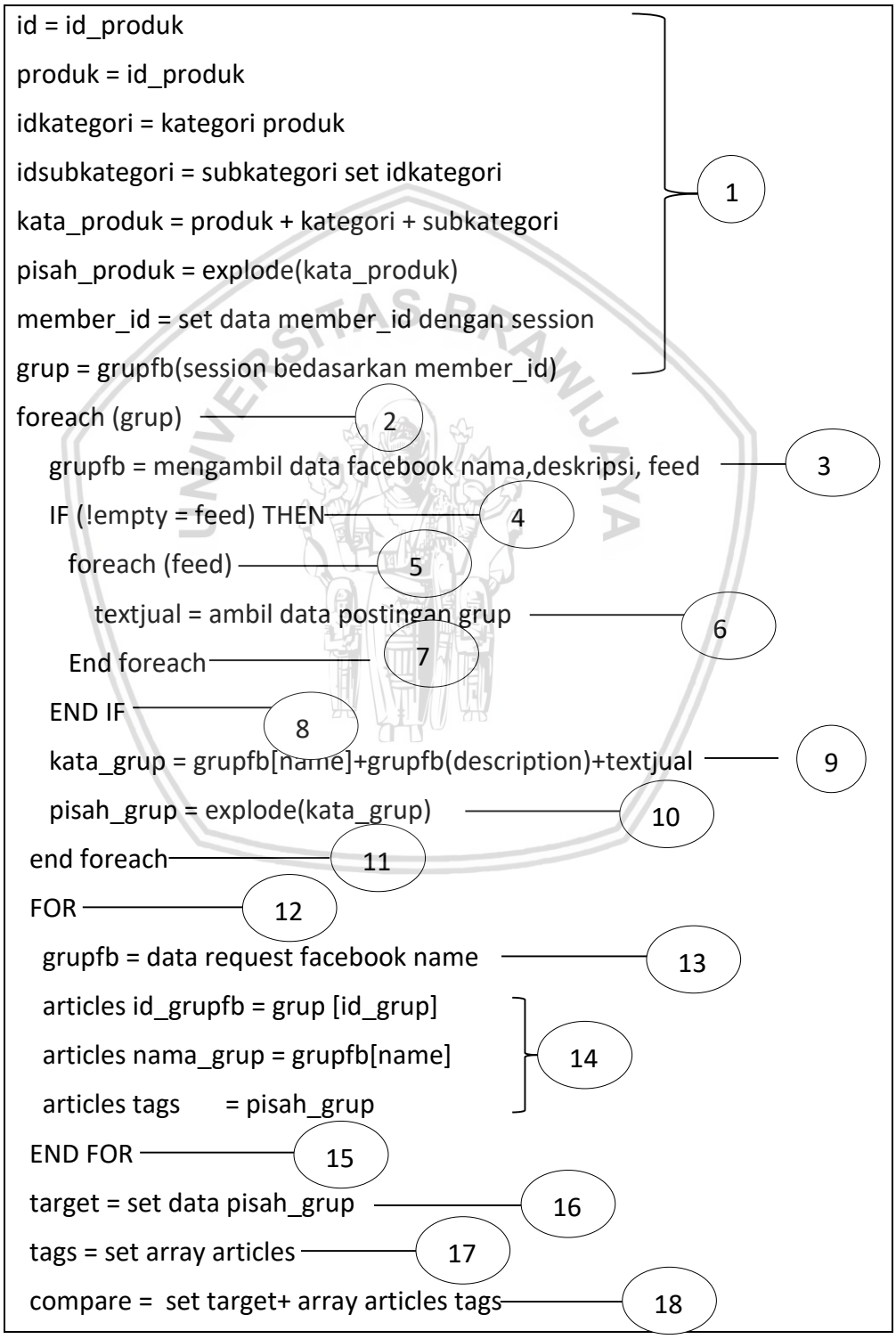
Tabel 6.2 Hasil Pengujian unit *operasi index*()

No Jalur	Prosedur Uji	Expected Result	Result	Status
1	Memanggil operasi index() kemudian melakukan *autentifikasi facebook* dengan *Graph* API	Mendapatkan *Data* akun *facebook* dan menyimpan ke *database* dan menampilkan	Mendapatkan *Data akun facebook* dan menyimpan ke *database* dan menampilkan	*Valid*
2	Memanggil operasi index() kemudian melakukan *autentifikasi facebook* dengan Graph API	Mendapatkan *Data* akun *facebook* dan menyimpan ke *database* tapi gagal menampilkan	Mendapatkan *Data akun facebook* dan menyimpan ke *database* tapi gagal menampilkan	*Valid*
3	Memanggil operasi index() kemudian melakukan *autentifikasi facebook* dengan *Graph* API	Tidak berhasil melakukan *autentifikasi facebook* dan gagal melakukan penyimpanan ke *database*	Tidak berhasil melakukan *autentifikasi facebook* dan gagal melakukan penyimpanan ke *database*	*Valid*

6.1.1.2 Pengujian unit *Cosinesimilarity*

- Nama Klas *(controller)* : Cosinesimilarity.php
- Nama Operasi : perhitungan()
- Proses Uji : Tabel 6.3

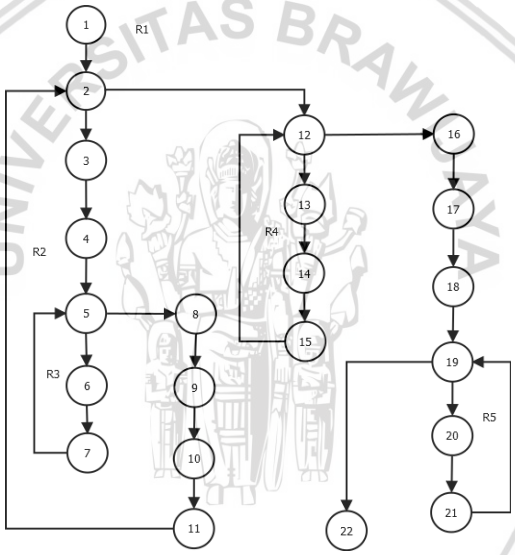
Tabel 6.3 Pengujian Unit *cosine similarity*



```
foreach (articles)                                              19
  ak = set arrray articles tagss + tags
  idgrup = set data articles id_grup                            20
  nama   = set data articles nama_grup
  score  = set data decimal ((katda_produk*kata_grup) kata_produk +
    kata_grup)/(kata_produk * kata_grup)/100
end foreach                                                     21
  tampilkan halaman cosinesimilarity                            22
```

Basis Path Testing

1. *Flow Graph*



Gambar 6.2 *Floiw Graph Operasi* perhitungan()

2. *Cyclomatic Complexity*
   a. V(G) = Jumlah Region = 5
   b. V(G) = E – N + 2 = 25– 22 + 2 = 5
   c. V(G) = P + 1 = 4+ 1 = 5
3. *Independent Path*
   a. Jalur 1: 1– 2 – 12 – 16 – 17 – 18 – 19 – 22
   b. Jalur 2: 1– 2 – 3 – 4 – 5 – 8 – 9 – 10 – 11 – 2 – 12 – 16 – 17 – 18 – 19 – 22
   c. Jalur 3: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 2 – 10 – 11 – 12 – 13 – 14 – 12 – 15 – 16 – 17 – 18 – 9 – 19 – 23 – 24 – 25 – 26 – 29

d. Jalur 4: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 2 – 10 – 11 – 12 – 13 – 14 – 12 – 15 – 16 – 17 – 18 – 9 – 19 – 20 – 21 – 22 – 19 – 23 – 24 – 25 – 26 – 29

e. Jalur 5: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 12 – 15 – 16 – 17 – 18 – 9 – 19 – 20 – 21 – 22 – 19 – 23 – 24 – 25 – 26 – 27 – 28 – 26 – 29

Tabel 6.4 Hasil Pengujian unit operasi perhitungan()

No Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1	Memanggil operasi perhituang*()* Dengan detail dari produk kemudian menampilkan halaman cosine	Menampilkan detail produk pada halaman *cosine similarity*	Menampilkan *detail* produk pada halaman *cosine similarity*	*Valid*
2	Memanggil operasi perhituang*()* Dengan detail dari produk kemudian mengirimkan data *grup facebook* menampilkan halaman cosine	Melakukan pemanggilan *data* grup facebook dengan *API facebook*	Melakukan pemanggilan *data* grup facebook dengan *API facebook*	*Valid*
3	Memanggil operasi perhituang*()* Dengan detail dari produk kemudian mengirimkan data *grup facebook* menampilkan halaman cosine	Melakukan pemanggilan *data message grup facebook*	Melakukan pemanggilan *data message grup facebook*	*Valid*
4	Memanggil operasi perhituang*()* Dengan *detail* dari produk kemudian mengirimkan data *id grup, nama grup, feed* dan juga detail produk	Mengirimkan data id *grup*, nama *grup*, *feed* dan *detail* produk	Mengirimkan *data* id *grup*, nama *grup*, *feed* dan *detail* produk	*Valid*

5	Memanggil operasi perhituang*()* Dengan detail dari produk kemudian melakukan perhitungan data *id grup, nama grup, feed* dan juga detail produk kemudian menampilkan pada halaman *cosine*	Melakukan perhitungan kemiripan dari data *grup facebook* dengan *detail* dari produk	Melakukan perhitungan kemiripan dari data *grup facebook* dengan *detail* dari produk	*Valid*

6.1.1.3 Pengujian unit *post facebook*

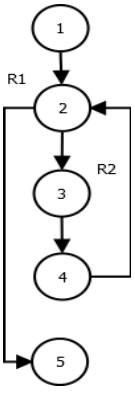
- Nama Klas *(controller)* : Cosinesimilarity.php
- Nama Operasi : postfacebook()
- Proses Uji : Tabel 6.5

Tabel 6.5 Pengujian Unit *Post facebook*

```
checkbox = mengambil input sesuai checkbox          (1)
foreach (checkbox)                                  (2)
   ambil data detail grup                           (3)
   post date ke grup facebook                       (3)
end foreach                                         (4)
tampil halaman daftar produk                        (5)
```

Basis Path Testing

1. *Flow Graph*



Gambar 6.3 *Floiw Graph Operasi postFacebook*()

2. *Cyclomatic Complexity*
   - V(G) = Jumlah Region = 2
   - V(G) = E – N + 2 = 5– 5 + 2 = 2
   - V(G) = P + 1 = 1+ 1 = 2
3. *Independent Path*
   - Jalur 1: 1 – 2 –5
   - Jalur 2: 1 – 2 – 3 – 4 – 2 – 5

Tabel 6.6 Hasil Pengujian unit operasi postfacebook()

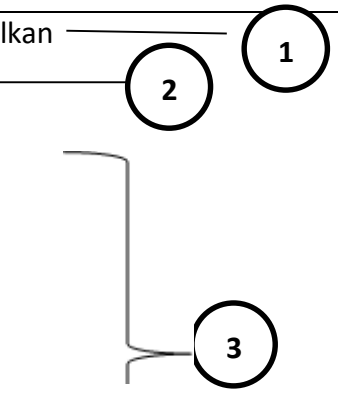
No Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1	Memanggil operasi postproduk*()* mengambil nilai dari variabel checkbox	Mengambil nilai *data* dari checkbox	Mengambil nilai *data* dari checkbox	*Valid*
2	Memanggil operasi postproduk *()* melakukan post produk ke *facebook* berupa *link* dan *message*	Mingirimkan *data detail* produk, deskripsi *post* dan *url* produk ke *grup facebok*	Mingirimkan *data* detail produk, deskripsi *post* dan *url* produk ke *grup facebok*	*Valid*

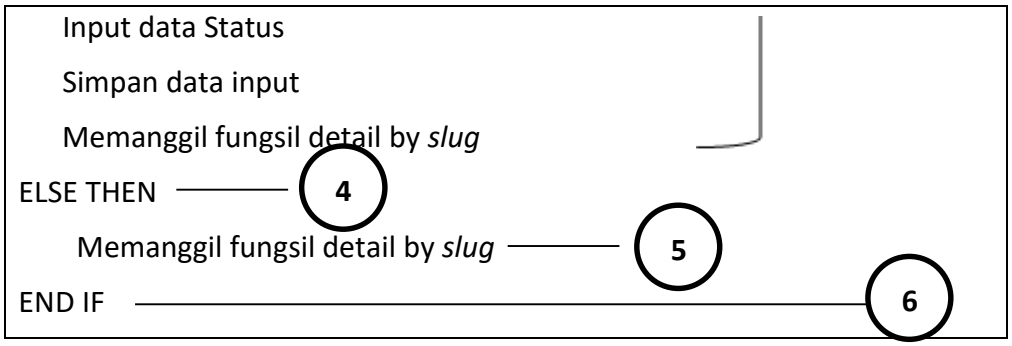
6.1.1.4 Pengujian unit *addcart*

- Nama Klas *(controller)* : KeranjangBelanja.php
- Nama Operasi : addcart ()
- Proses Uji : Tabel 6.7

Tabel 6.7 Pengujian unit Addcart

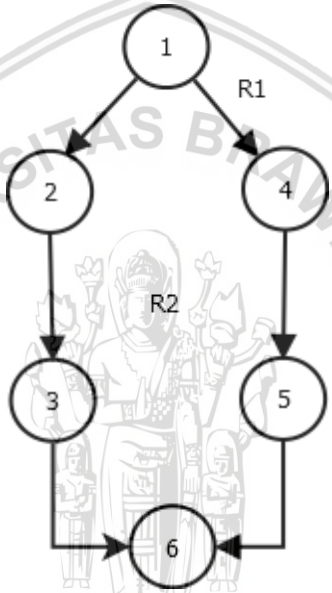
```
Slug = mengambil slug dari produk yang ditampilkan
IF(submit) THEN
    Input  data Qty
    Input data Seller
    Input data Member_Id
    Input data Gambar
    Input data Gambar
```





Basis Path Testing

1. *Flow Graph*



Gambar 6.4 *Flow graph* Operasi addcart

2. *Cyclomatic Complexity*
   - V(G) = Jumlah Region = 2
   - V(G) = E – N + 2 = 6– 6 + 2 = 2
   - V(G) = P + 1 = 1+ 1 = 2
3. *Independent Path*
   - Jalur 1: 1 – 4 – 5 – 6
   - Jalur 2: 1 – 2 – 3 – 6

Tabel 6.8 Hasil Pengujian unit operasi *Addcart*()

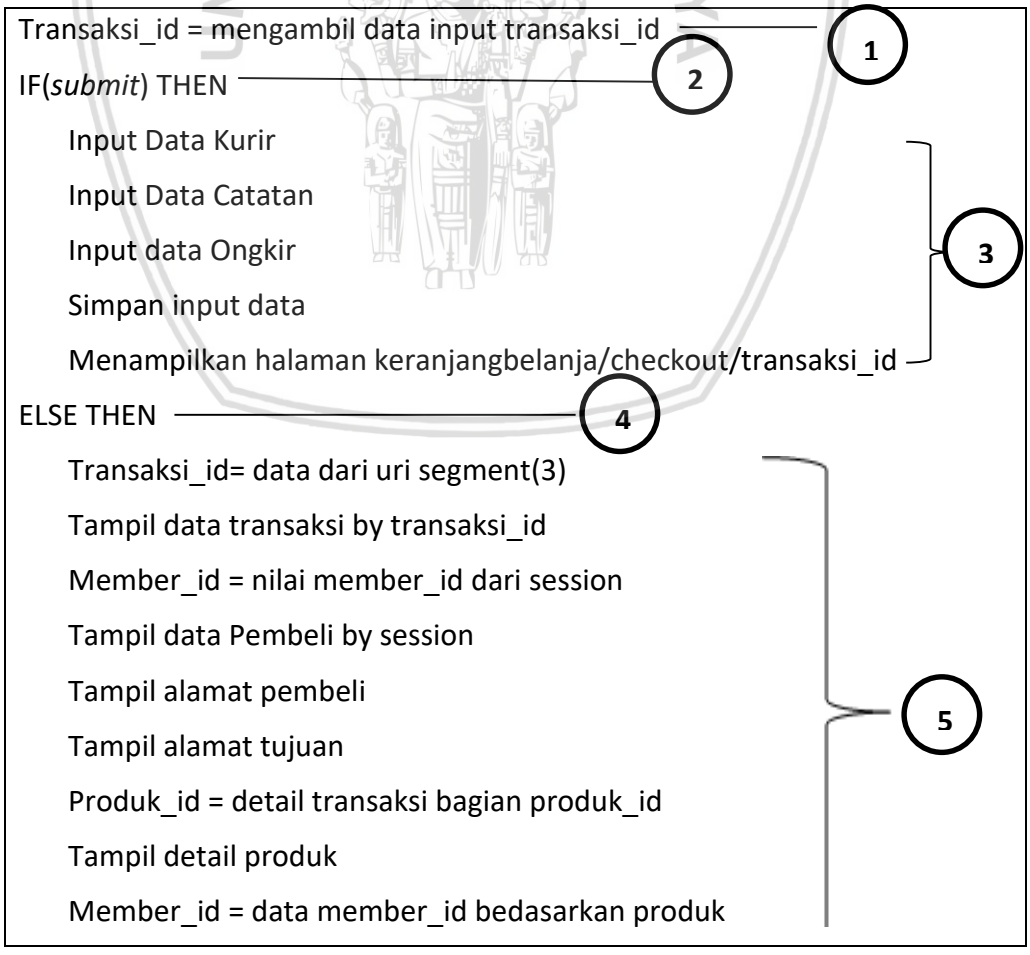
No Jalur	Prosedur Uji	*Expected Result*	*Result*	Status

1	Memanggil operasi *addcart()* dengan kondisi submit maka akan *insert* data ke *database*	Menambahkan item keranjang belanja	Menambah item keranjang belanja	*Valid*
2	Memanggil operasi *addcart ()* dengan kondisi != submit maka akan menampilkan detail produk	Menampilkan halaman view *detail* produk	Menampilkan halaman view *detail* produk	*Valid*

6.1.1.5 Pengujian unit *checkout*

- Nama Klas *(controller)* : KeranjangBelanja.php
- Nama Operasi : checkout ()
- Proses Uji : Tabel 6.9

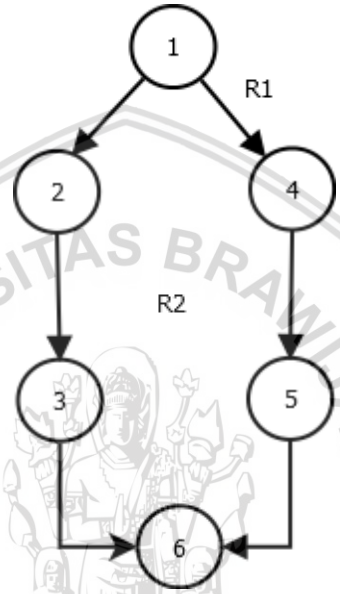
Tabel 6.9 Pengujian Unit checkout



Tampil detail penjual Tampil data pengiriman Tampil alamat penjual by penjual_id Tampil data alamat penjual END IF	6

Basis Path Testing

1. *Flow Graph*



Gambar 6.5 *Flow Graph* Operasi checkout

2. *Cyclomatic Complexity*
   - V(G) = Jumlah Region = 2
   - V(G) = E – N + 2 = 6– 6 + 2 = 2
   - V(G) = P + 1 = 1+ 1 = 2
3. *Independent Path*
   - Jalur 1: 1 – 4 – 5
   - Jalur 2: 1 – 2 – 3 – 5

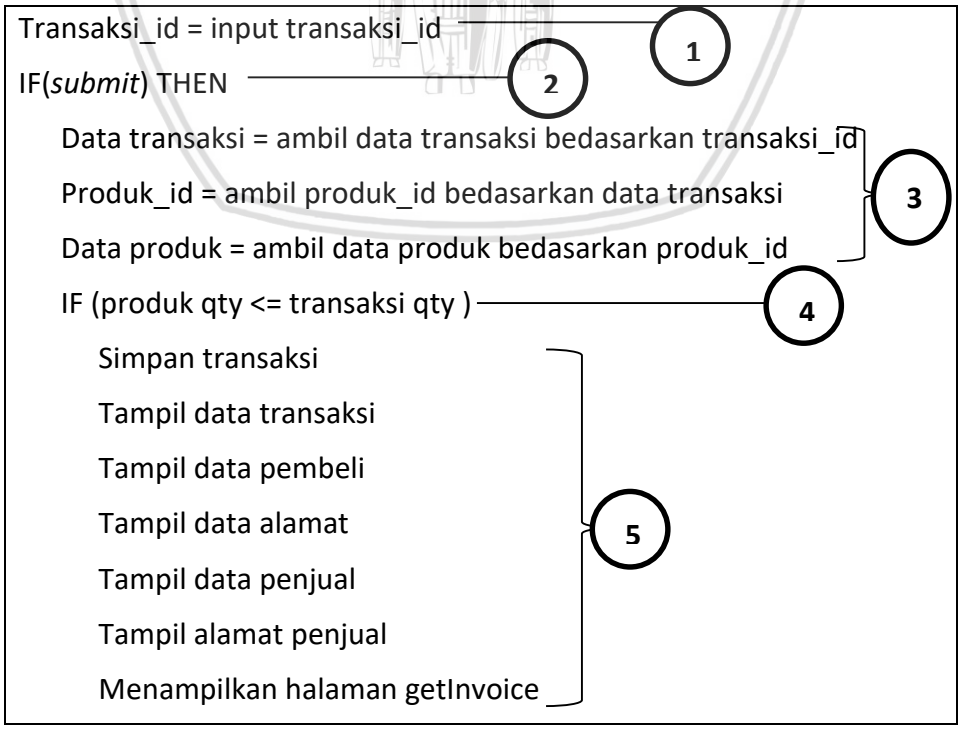
Tabel 6.10 Hasil Pengujian unit operasi *checkout()*

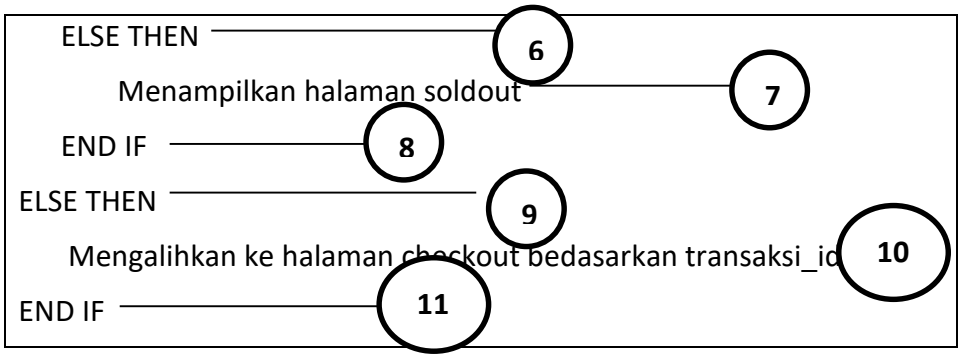
No Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1	Memanggil operasi *checkout()* dalam kondisi submit untuk melakukan update *item* belanja	Melakukan *Update item* keranjang belanja	Melakukan *Update item* keranjang belanja	*Valid*
2	Memanggil operasi *checkout ()* dengan kondisi != submit menampilkan detail dari item belanja	Menampilkan halaman view detail item keranjang belanja	Menampilkan halaman *view detail* item keranjang belanja	*Valid*

6.1.1.6 Pengujian unit get*invoice*

- Nama Klas *(controller)* : KeranjangBelanja.php
- Nama Operasi : getInvoice ()
- Proses Uji : Tabel 6.11

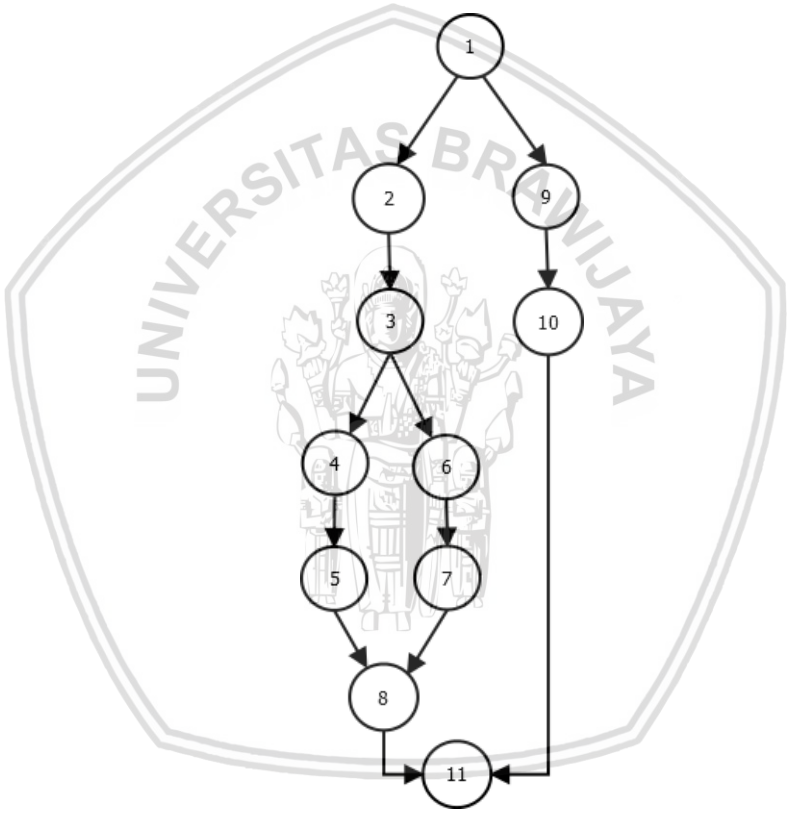
Tabel 6.11 Pengujian unit dapatkan kode tagihan





Basis Path Testing

1. *Flow Graph*



Gambar 6.6 *Flow Graph Operasi getInvoice*()

2. *Cyclomatic Complexity*
   a. V(G) = Jumlah Region = 3
   b. V(G) = E – N + 2 = 12– 11 + 2 = 3
   c. V(G) = P + 1 = 2+ 1 = 3
3. *Independent Path*
   a. Jalur 1: 1– 2 – 3 – 4 – 5 – 8 – 11
   b. Jalur 2: 1– 2 – 3 – 6 – 7 – 8 – 11

c. Jalur 3: 1 – 9 – 10 – 11

Tabel 6.12 Hasil Pengujian unit operasi index()

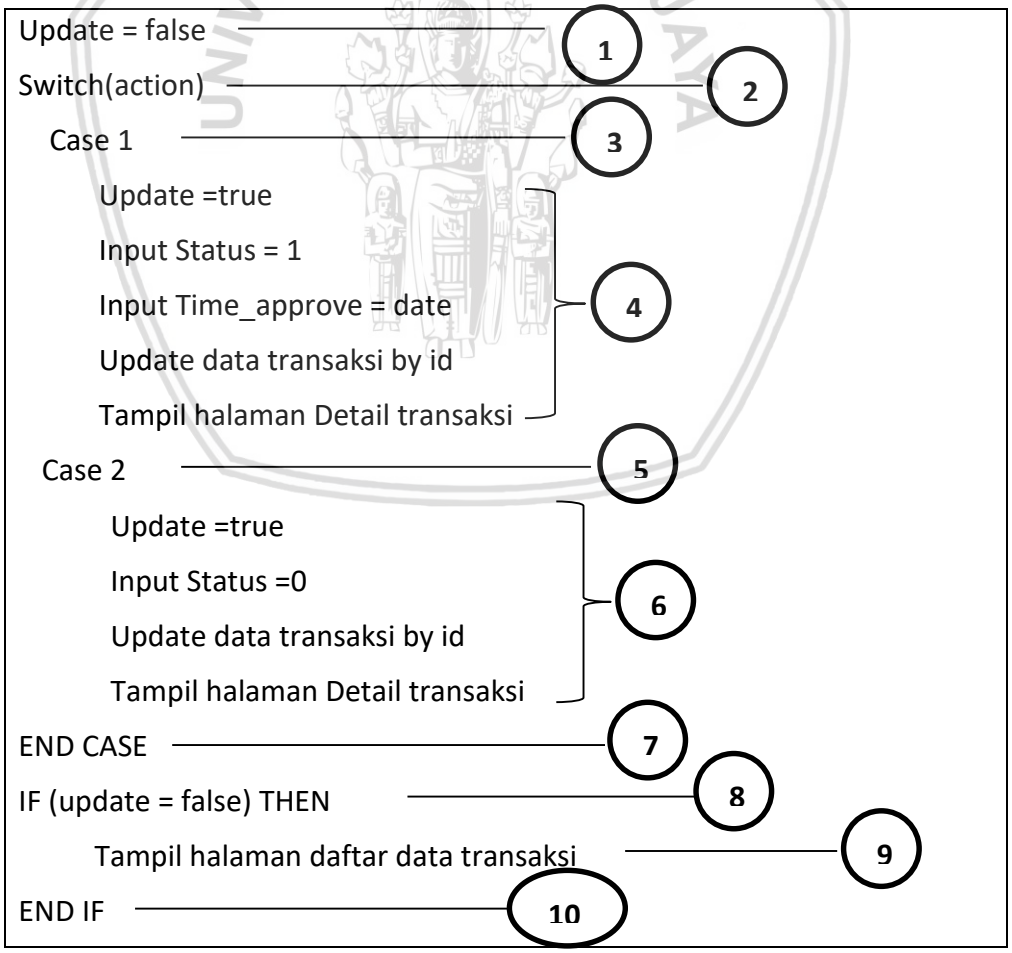
No Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1	Memanggil operasi getInvoice() menghasilkan kode tagihan pembayaran sesuai dengan jenis pembayaran	Jika pembelian barang tidak lebih dari jumlah ketersedian produk maka akan menyimpan hargasatuan, subtotal, ongkos kirim, status, jenis pembayaran dan menghasilkan *invoice* atau kode tagihan	Jika pembelian barang tidak lebih dari jumlah ketersedian produk maka akan menyimpan hargasatuan, subtotal, ongkos kirim, status, jenis pembayaran dan menghasilkan *invoice* atau kode tagihan	*Valid*
2	Memanggil operasi getInvoice() menghasilkan kode tagihan pembayaran sesuai dengan jenis pembayaran	Jika pembelian barang lebih besar dari jumlah ketersedian produk atau produk tidak tersedia maka akan menampilkan halaman *sold out*	Jika pembelian barang lebih besar dari jumlah ketersedian produk atau produk tidak tersedia maka akan menampilkan halaman *sold out*	*Valid*

3	Memanggil operasi getInvoice() menghasilkan kode tagihan pembayaran sesuai dengan jenis pembayaran	Jika tidak melakukan submit maka akan dialihakan ke halaman cheokout bedasarkan transaksi_id	Jika tidak melakukan submit maka akan dialihakan ke halaman cheokout bedasarkan transaksi_id	*Valid*

6.1.1.7 Pengujian unit Konfirmasi transaksi produk admin

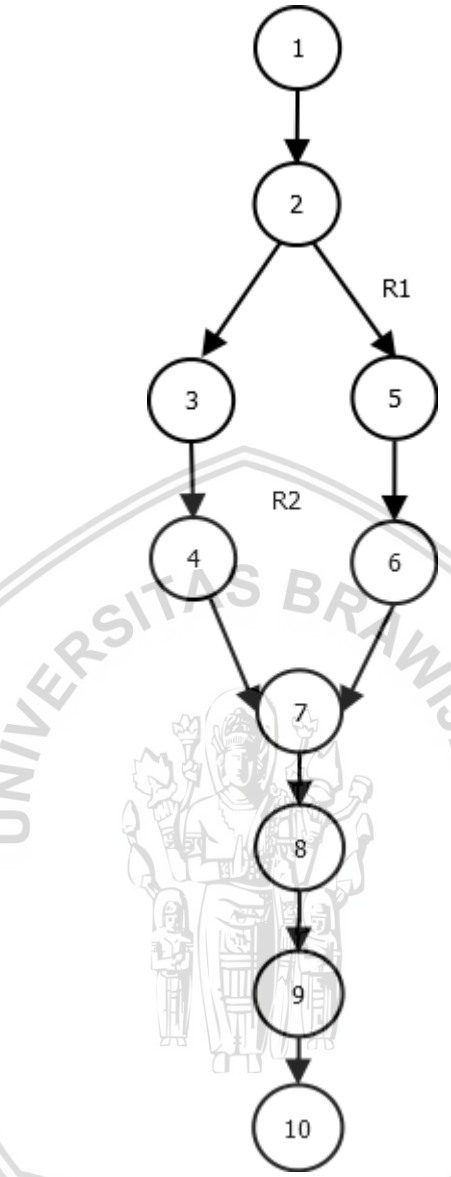
- Nama Klas *(controller)* : DataTransaksi.php
- Nama Operasi : Update ()
- Proses Uji : Tabel 6.13

Tabel 6.13 Pengujian Unit konfirmasi transaksi produk admin



Basis Path Testing

1. *Flow Graph*



Gambar 6.7 *Flow graph operasi update()*

2. *Cyclomatic Complexity*
   - V(G) = Jumlah Region = 2
   - V(G) = E – N + 2 = 10– 10 + 2 = 2
   - V(G) = P + 1 = 1+ 1 = 2
3. *Independent Path*
   - Jalur 1: 1 – 2 – 5 – 6 – 7 – 8 – 9 – 10
   - Jalur 2: 1 – 2 – 3 – 4 –7 – 8 – 9 – 10

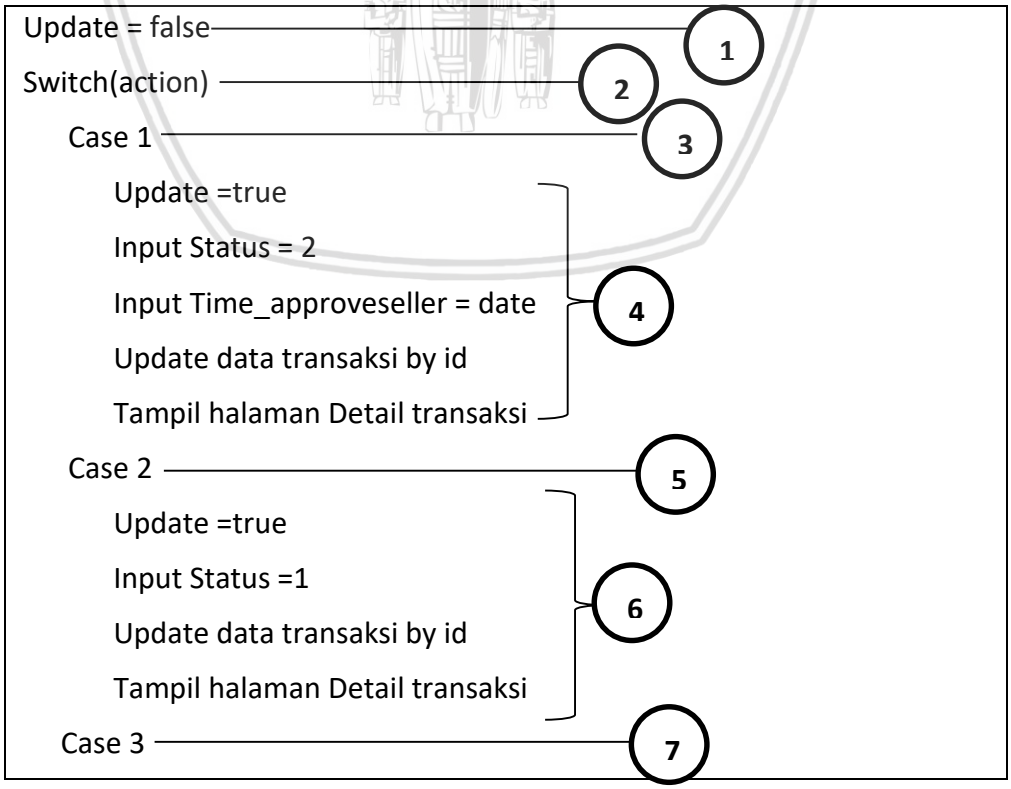
Tabel 6.14 Hasil Pengujian unit operasi *update*()

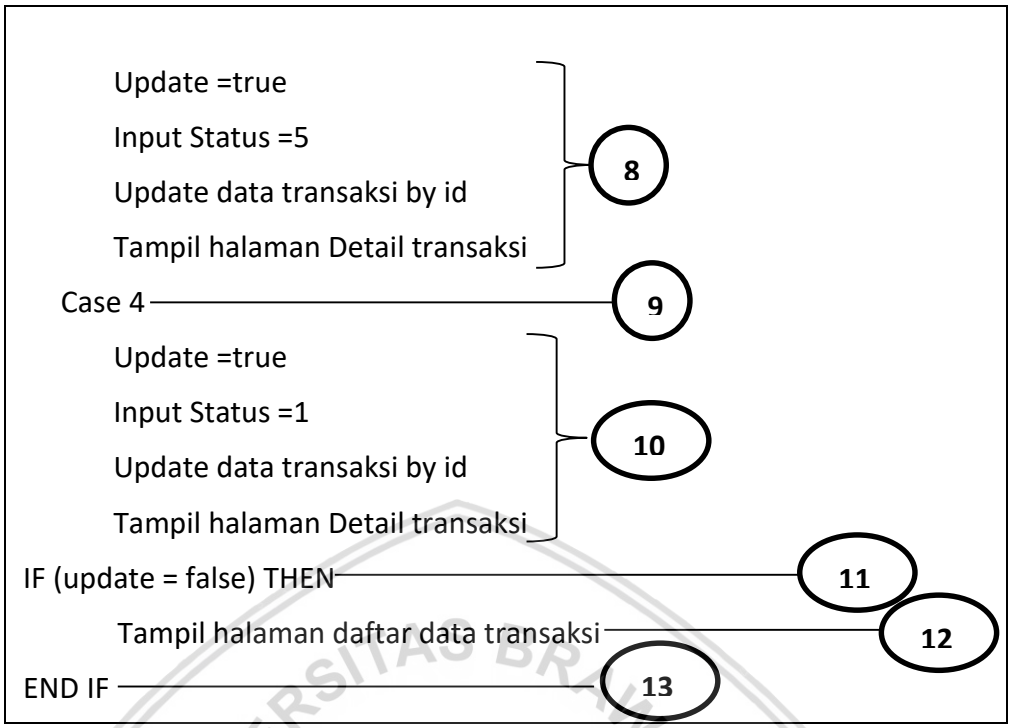
No Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1	Memanggil operasi update*()* dalam kondisi case transaksi diterima / ditolak admin	Melakukan konfirmasi transaksi	Melakukan konfirmasi transaksi	*Valid*
2	Memanggil operasi update*()* dalam kondisi case transaksi diterima/ditolak admin	Melakukan penolakan transaksi	Melakukan penolakan transaksi	*Valid*

6.1.1.8 Pengujian unit konfirmasi transaksi produk penjual

- Nama Klas *(controller)* : Transaksi.php
- Nama Operasi : update()
- Proses Uji : Tabel 6.15

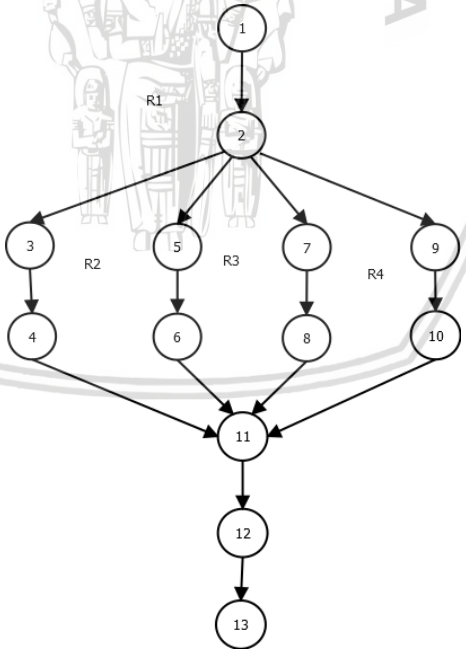
Tabel 6.15 Pengujian Unit konfirmasi transaksi produk penjual





Basis Path Testing

1. *Flow Graph*



Gambar 6.8 *Flow Graph Operasi update()*

2. *Cyclomatic Complexity*

   a. V(G) = Jumlah Region = 4

   b. V(G) = E – N + 2 = 15– 13 + 2 = 4

c. V(G) = P + 1 = 3+ 1 = 4

3. *Independent Path*

a. Jalur 1: 1 – 2 – 3 – 4 – 11 – 12 – 13

b. Jalur 2: 1 – 2 – 5 – 6 – 11 – 12 – 13

c. Jalur 3: 1 – 2 – 7 – 8 – 11 – 12 – 13

d. Jalur 4: 1 – 2 – 9 – 10 – 11 – 12 – 13

Tabel 6.16 Hasil Pengujian unit operasi update()

No Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1	Memanggil operasi update*()* dalam kondisi case transaksi diterima / ditolak penjual	Melakukan konfirmasi transaksi	Melakukan konfirmasi transaksi	*Valid*
2	Memanggil operasi update*()* dalam kondisi case transaksi diterima/ditolak penjual	Melakukan batal terima transaksi	Melakukan batal terima transaksi	*Valid*
3	Memanggil operasi update*()* dalam kondisi case transaksi diterima/ditolak penjual	Melakukan penolakan transaksi	Melakukan penolakan transaksi	*Valid*
4	Memanggil operasi update*()* dalam kondisi case transaksi diterima/ditolak penjual	Melakukan pembatalan penolokan transaksi	Melakukan pembatalan penolokan transaksi	*Valid*

6.1.2 Pengujian Integrasi

Pada pengujian ini akan menguji mengenai keterkaitan atau hubungan antar method ataupun klas. Dalam penelitian ini pengujian integrasi dilakukan berdasarkan operasi atau fungsi yang telah diuji pada pengujian unit sebelumnya.

6.1.2.1 Pengujian integrasi logoutfb

Pada *method* logout pada klas koneksifb akan memanggil method index yang sama-sama berada di klas koneksifb. Algoritme dan pembentukan node dapat dilihat pada tabel 6.17 sedangkan gambar 6.9 menunjukkan *flow graph* dari method getListProduk.

Tabel 6.17 Pembentukan node algoritme logoutfb

Hapus session facebook Lepas data session Memanggil fungsi index	1

Basis Path Testing

1. *Flow Graph*

Gambar 6.9 *Flow Graph* algoritme getListProduk

2. *Cyclomatic Complexity*
   - V(G) = E – N + 2
   - V(G) = 0 – 1 + 2 = 1
   - V(G) = 0 + 1 = 0 + 1 = 1
3. *Independent Path*
   a. Jalur 1: 1

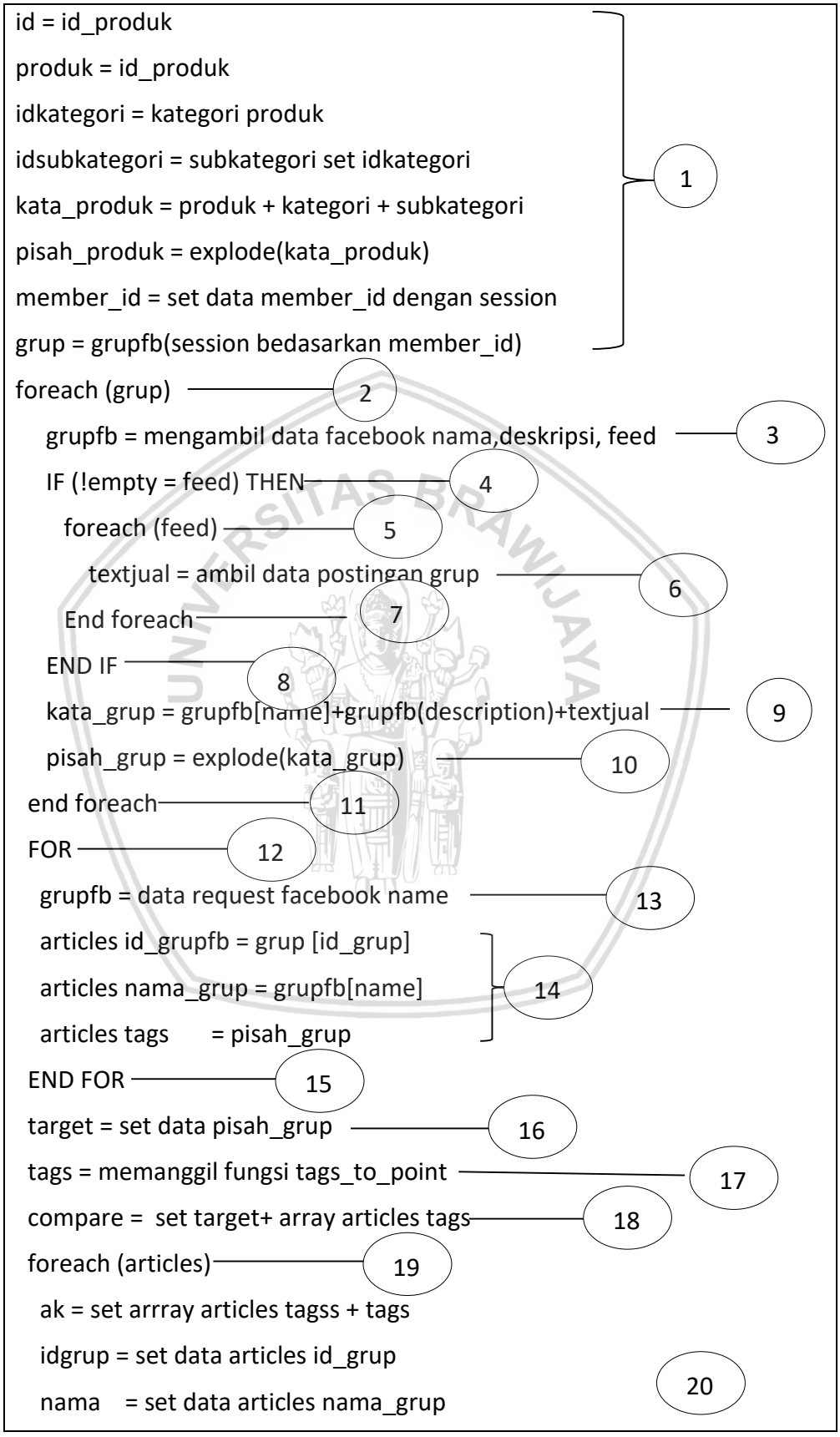
Tabel 6.18 Hasil Pengujian integrasi algoritme getListProduk

No Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1	Memanggil fungsi logoutfb	Menghapus session facebook dan memanggil fungsi index pada klas koneksifb	Menghapus session facebook dan memanggil fungsi index pada klas koneksifb	*Valid*

6.1.2.2 Pengujian integrasi perhitungan

Pada method perhitungan pada klas Cosinesimilarity akan memanggil method cosine yang sama-sama berada di klas Cosinesimilarity dan juga method tags_to_point pada klas cosinesimilarity. Algoritme dan pembentukan node dapat dilihat pada tabel 6.19 sedangkan gambar 6.10 menunjukan flow graph dari method getListProduk.

Tabel 6.19 Pembetukan *node* algoritme perhitungan



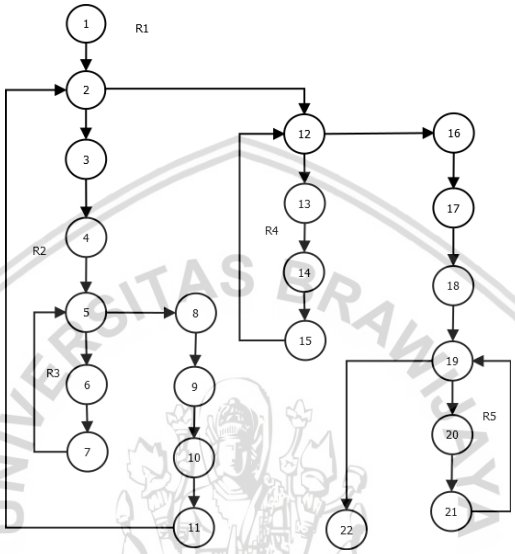
score = set data decimal(memanggil fungsi cosine)/100

end foreach — 21

tampilkan halaman cosinesimilarity — 22

Basis Path Testing

4. Flow Graph



Gambar 6.10 *Floiw Graph Operasi* perhitungan()

5. Cyclomatic Complexity

- V(G) = Jumlah Region = 5
- V(G) = E – N + 2 = 25– 22 + 2 = 5
- V(G) = P + 1 = 4+ 1 = 5

6. Independent Path

a. Jalur 1: 1– 2 – 12 – 16 – 17 – 18 – 19 – 22

b. Jalur 2: 1– 2 – 3 – 4 – 5 – 8 – 9 – 10 – 11 – 2 – 12 – 16 – 17 – 18 – 19 – 22

c. Jalur 3: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 2 – 10 – 11 – 12 – 13 – 14 – 12 – 15 – 16 – 17 – 18 – 9 – 19 – 23 – 24 – 25 – 26 – 29

d. Jalur 4: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 2 – 10 – 11 – 12 – 13 – 14 – 12 – 15 – 16 – 17 – 18 – 9 – 19 – 20 – 21 – 22 – 19 – 23 – 24 – 25 – 26 – 29

e. Jalur 5: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 12 – 15 – 16 – 17 – 18 – 9 – 19 – 20 – 21 – 22 – 19 – 23 – 24 – 25 – 26 – 27 – 28 – 26 – 29

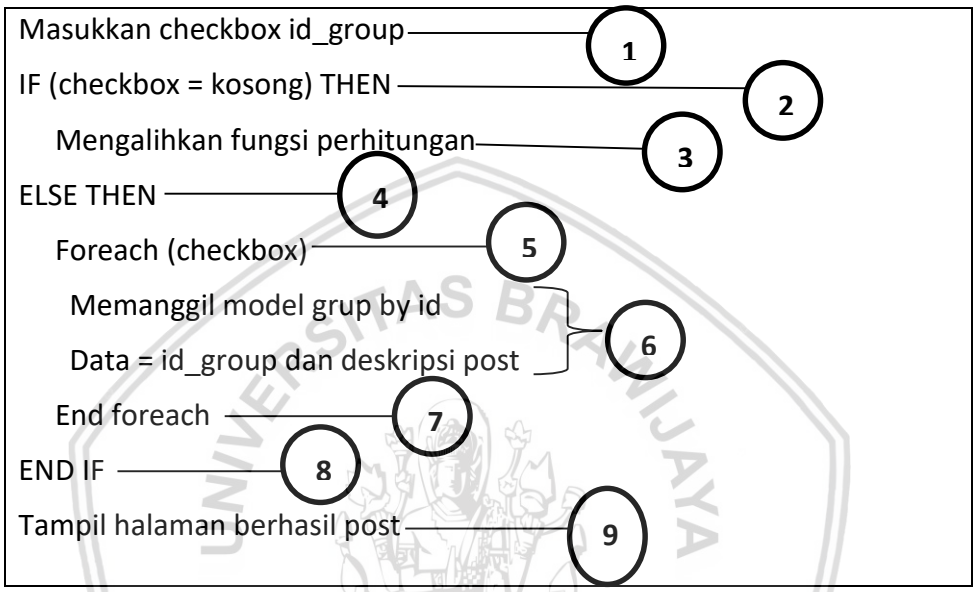
Tabel 6.20 Hasil Pengujian unit operasi perhitungan()

No Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1	Memanggil operasi perhituang*()* Dengan detail dari produk kemudian menampilkan halaman cosine	Menampilkan detail produk pada halaman *cosine similarity*	Menampilkan *detail* produk pada halaman *cosine similarity*	*Valid*
2	Memanggil operasi perhituang*()* Dengan detail dari produk kemudian mengirimkan data *grup facebook* menampilkan halaman cosine	Melakukan pemanggilan *data* grup facebook dengan *API facebook*	Melakukan pemanggilan *data* grup facebook dengan *API facebook*	*Valid*
3	Memanggil operasi perhituang*()* Dengan detail dari produk kemudian mengirimkan data *grup facebook* menampilkan halaman cosine	Melakukan pemanggilan *data message grup facebook*	Melakukan pemanggilan *data message grup facebook*	*Valid*
4	Memanggil operasi perhituang*()* Dengan *detail* dari produk kemudian mengirimkan data *id grup, nama grup, feed* dan juga detail produk	Mengirimkan data id *grup*, nama *grup*, *feed* dan *detail* produk	Mengirimkan *data* id *grup*, nama *grup*, *feed* dan *detail* produk	*Valid*
5	Memanggil operasi perhituang*()* Dengan detail dari produk kemudian melakukan perhitungan data *id grup, nama grup, feed* dan juga detail produk kemudian menampilkan pada halaman *cosine*	Melakukan perhitungan kemiripan dari data *grup facebook* dengan *detail* dari produk	Melakukan perhitungan kemiripan dari data *grup facebook* dengan *detail* dari produk	*Valid*

6.1.2.3 Pengujian integrasi *post*

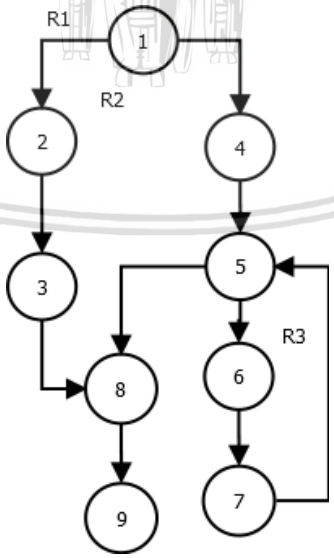
Pada *method* post pada klas Cosinesimilarity akan memanggil method perhitungan yang sama-sama berada di klas Cosinesimilarity. Algoritme dan pembentukan node dapat dilihat pada tabel 6.21 sedangkan gambar 6.11 menunjukkan *flow graph* dari method post.

Tabel 6.21 Pembetukan *node* algoritme post



Basis Path Testing

1. Flow Graph



Gambar 6.11 *Flow graph* Operasi *post*

2. Cyclomatic Complexity

a. V(G) = Jumlah Region = 3

b. V(G) = E – N + 2 = 10– 9 + 2 = 3

c. V(G) = P + 1 = 2+ 1 = 3

3. Independent Path

a. Jalur 1: 1 – 2 – 3 – 8 – 9

b. Jalur 2: 1 – 4 – 5 – 8 – 9

c. Jalur 3: 1 – 4 – 5 – 6 – 7 – 5 – 8 – 9

Tabel 6.22 Hasil Pengujian unit operasi *post*()

No Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1	Memanggil operasi post	Jika chekbox bernilai kosong maka halaman akan tetap pada halaman cosine	Jika chekbox bernilai kosong maka halaman akan tetap pada halaman cosine	*Valid*
2	Memanggil operasi post	Jika chekbox bernilai id_group maka halaman akan dialihkan ke halaman berhasilposting	Jika chekbox bernilai id_group maka halaman akan dialihkan ke halaman berhasilposting	*Valid*
3	Memanggil operasi post	Jika chekbox bernilai id_group maka akan dilakukan posting ke grup *facebook* dan dialihkan ke halaman berhasil posting	Jika chekbox bernilai id_group maka akan dilakukan posting ke grup *facebook* dan dialihkan ke halaman berhasil posting	*Valid*

6.1.2.4 Pengujian integrasi addcart

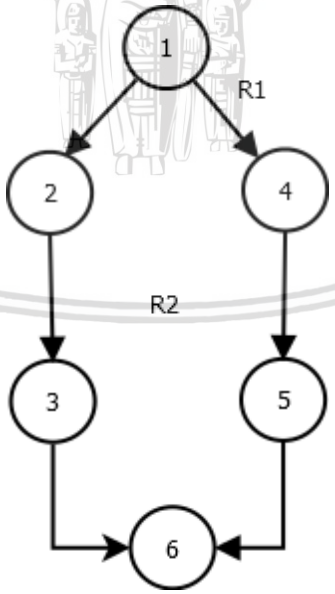
Pada *method addcart* pada klas keranjangbelanja akan memanggil method *detail* yang berada di klas produk. Algoritme dan pembentukan node dapat dilihat pada tabel 6.22 sedangkan gambar 6.12 menunjukkan *flow graph* dari method post.

Tabel 6.23 Pembetukan *node* algoritme *Addcart*

Slug = mengambil slug dari produk yang ditampilkan	1
IF(*submit*) THEN	2
Input data Qty	3
Input data Seller	
Input data Member_Id	
Input data Gambar	
Input data Gambar	
Input data Status	
Simpan data input	
Memanggil fungsil detail by *slug*	
ELSE THEN	4
Memanggil fungsil detail by *slug*	5
END IF	6

Basis Path Testing

4. Flow Graph



Gambar 6.12 *Flow graph* Operasi addcart

5. Cyclomatic Complexity

a. V(G) = Jumlah Region = 2

b. V(G) = E – N + 2 = 6– 6 + 2 = 2

c. V(G) = P + 1 = 1+ 1 = 2

6. Independent Path

a. Jalur 1: 1 – 4 – 5 – 6

b. Jalur 2: 1 – 2 – 3 – 6

Tabel 6.24 Hasil Pengujian unit operasi *Addcart*()

No Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1	Memanggil operasi *addcart()* dengan kondisi submit maka akan *insert* data ke *database*	Menambahkan item keranjang belanja	Menambah item keranjang belanja	*Valid*
2	Memanggil operasi *addcart ()* dengan kondisi != submit maka akan menampilkan detail produk	Menampilkan halaman view *detail* produk	Menampilkan halaman view *detail* produk	*Valid*

6.2 Pengujian *Black-box*

Pada pengujian *black-box* ini menggunakan *Equivalence Partitioning.* Metode *Equivalence Partitioning* akan ujikan pada 14 kebutuhan yang telah ditetapkan. Kebutuhan yang diujikan meliputi masuk admin, tambah data produk kategori, perbarui produk kategori, tambah user admin, daftar, masuk, tambah produk, perbarui produk, bagikan info produk, perbarui *profil member,* tambah item keranjang belanja, *checkout,* tambah alamat dan tambah testimoni. Seperti pada hasil tabel berikut.

6.2.1 Pengujian Validasi Masuk admin

Tabel 6.25 Kasus Uji Validasi Masuk admin

Nomor Kasus Uji Validasi	**VAL_01**			
Nama Kasus Uji	**Kasus uji validasi masuk admin**			
NO	**Test Case**	**Input**	**Expected**	**Result Test**
1	Admin menginputkan username dan password string	Username = admin Password = admin	Berhasil login	Valid

Nomor Kasus Uji Validasi	VAL_01			
Nama Kasus Uji	Kasus uji validasi masuk admin			
NO	**Test Case**	**Input**	**Expected**	**Result Test**
2	Admin menginputkan username null dan password string	Username = null Password = admin	Gagal login	Valid
3	Admin menginputkan username string dan password null	Username = admin Password = null	Gagal login	Valid

6.2.2 Pengujian Validasi Tambah Data Produk Kategori

Tabel 6.26 Kasus Uji Validasi Tambah Data Produk Kategori

Nomor Kasus Uji Validasi	VAL_02			
Nama Kasus Uji	Kasus uji validasi tambah data produk kategori			
NO	**Test Case**	**Input**	**Expected**	**Result Test**
1	Admin menginputkan Nama kategori string dan Icon kategori char	Nama kategori = elektronik Icon kategori = fa-bolt	Berhasil tambah kategori	Valid
2	Admin menginputkan Nama kategori string dan Icon kategori kosong	Nama kategori = elektronik Icon kategori = null	Berhasil tambah kategori	Valid
3	Admin menginputkan Nama kategori kosong dan Icon kategori char	Nama kategori = null Icon kategori = fa-bolt	Gagal tambah kategori	Valid

6.2.3 Pengujian Validasi Perbarui Data Produk Kategori

Tabel 6.27 Kasus Uji Validasi Perbarui data Produk kategori

Nomor Kasus Uji Validasi	**VAL_03**			
Nama Kasus Uji	**Kasus Uji Validasi Perbarui Data Produk Kategori**			
NO	**Test Case**	**Input**	**Expected**	**Result Test**
1	Admin menginputkan Nama kategori string dan Icon kategori char	Nama kategori = elektronik Icon kategori = fa-bolt	Berhasil perbarui kategori	Valid
2	Admin menginputkan Nama kategori string dan Icon kategori kosong	Nama kategori = elektronik Icon kategori = null	Berhasil perbarui kategori	Valid
3	Admin menginputkan Nama kategori kosong dan Icon kategori char	Nama kategori = null Icon kategori = fa-bolt	Gagal perbarui kategori	Valid

6.2.4 Pengujian Validasi Tambah User Admin

Tabel 6.28 Kasus Uji Validasi Tambah User Admin

Nomor Kasus Uji Validasi	**VAL_04**			
Nama Kasus Uji	**Kasus Uji Validasi Tambah User Admin**			
NO	**Test Case**	**Input**	**Expected**	**Result Test**
1	Admin menginputkan Username dengan nilai string, email dengan nilai char dan password dengan nilai string	Username = wildan Email = wildan@mail.com Password = wildan	Berhasil tambah User admin baru	Valid
2	Admin menginputkan Username dengan nilai string, email dengan nilai string dan password dengan nilai string	Username = wildan Email = wildanmail	Gagal tambah User Admin	Valid

Nomor Kasus Uji Validasi	VAL_04			
Nama Kasus Uji	Kasus Uji Validasi Tambah User Admin			
NO	**Test Case**	**Input**	**Expected**	**Result Test**
		Password = wildan		
3	Admin menginputkan Username dengan nilai kosong, email dengan nilai char dan password dengan nilai string	Username = null Email = wildan@mail.com Password = wildan	Gagal tambah User Admin	Valid

6.2.5 Pengujian Validasi Daftar

Tabel 6.29 Kasus Uji Validasi Daftar

Nomor Kasus Uji Validasi	VAL_05			
Nama Kasus Uji	Kasus Uji Validasi Daftar			
NO	**Test Case**	**Input**	**Expected**	**Result Test**
1	User menginputkan Nama dengan nilai string, email dengan nilai char, username dengan nilai string, password dengan nilai string	Nama = wildanmukafi, Email = wildan@mail.com, username = wildanmkf, password = wildan	Berhasil daftar member	Valid
2	User menginputkan Nama dengan nilai string, email dengan nilai numerik, username dengan nilai string, password dengan nilai string	Nama = wildanmukafi, Email = 12345, username = wildanmkf, password = wildan	Gagal daftar member	Valid
3	User menginputkan Nama dengan nilai string, email dengan nilai numerik, username kosong, password dengan nilai string	Nama = wildanmukafi, Email = wildan@mail.com, username =	Gagal daftar member	Valid

Nomor Kasus Uji Validasi		VAL_05		
Nama Kasus Uji		Kasus Uji Validasi Daftar		
NO	**Test Case**	**Input**	**Expected**	**Result Test**
		wildanmkf, password = null		

6.2.6 Pengujian Validasi Masuk

Tabel 6.30 Kasus Uji Validasi Masuk

Nomor Kasus Uji Validasi		VAL_06		
Nama Kasus Uji		Kasus Uji Validasi Masuk		
NO	**Test Case**	**Input**	**Expected**	**Result Test**
1	User menginputkan Username dengan nilai string dan password dengan nilai string	Username = wildan, password = wildan	Berhasil masuk member	Valid
2	User menginputkan Username kosong dan password dengan nilai string	Username = null, password = wildan	Gagal masuk member	Valid
3	User menginputkan Username dengan nilai string dan password kosong	Username = wildan, password = null	Gagal masuk member	Valid

6.2.7 Pengujian Validasi Tambah Produk

Tabel 6.31 Kasus Uji Validasi Tambah Produk

Nomor Kasus Uji Validasi		VAL_07		
Nama Kasus Uji		Kasus Uji Validasi Tambah Produk		
NO	**Test Case**	**Input**	**Expected**	**Result Test**
1	Member menginputkan judul dengan string, deskripsi dengan nilai char, harga dengan nilai numerik, berat dengan nilai numerik, jumlah dengan nilai numerik,	Judul = kulkas, deskripsi = kulkas dua pintu, harga = 10000, berat = 10, jumlah = 2	Berhasil Tambah produk baru	Valid

Nomor Kasus Uji Validasi	VAL_07			
Nama Kasus Uji	Kasus Uji Validasi Tambah Produk			
NO	**Test Case**	**Input**	**Expected**	**Result Test**
2	Member menginputkan judul dengan string, deskripsi dengan nilai char, harga dengan nilai string, berat dengan nilai numerik, jumlah dengan nilai numerik,	Judul = kulkas, deskripsi = kulkas dua pintu, harga = seribu, berat = 10, jumlah = 2	Gagal Tambah produk baru	Valid
3	Member menginputkan judul kosong, deskripsi dengan nilai char, harga dengan nilai numerik, berat dengan nilai numerik, jumlah dengan nilai numerik,	Judul = null, deskripsi = kulkas dua pintu, harga = 1000, berat = 10, jumlah = 2	Gagal Tambah produk baru	Valid

6.2.8 Pengujian Validasi Perbarui Produk

Tabel 6.32 Kasus Uji Validasi Perbarui Produk

Nomor Kasus Uji Validasi	VAL_08			
Nama Kasus Uji	Kasus Uji Validasi Perbarui Produk			
NO	**Test Case**	**Input**	**Expected**	**Result Test**
1	Member menginputkan judul dengan string, deskripsi dengan nilai char, harga dengan nilai numerik, berat dengan nilai numerik, jumlah dengan nilai numerik,	Judul = kulkas, deskripsi = kulkas dua pintu, harga = 10000, berat = 10, jumlah = 2	Berhasil perbarui produk	Valid
2	Member menginputkan judul dengan string, deskripsi dengan nilai char, harga kosong, berat dengan nilai numerik, jumlah dengan nilai numerik,	Judul = kulkas, deskripsi = kulkas dua pintu, harga = null, berat = 10, jumlah = 2	Gagal perbarui produk	Valid

Nomor Kasus Uji Validasi		VAL_08		
Nama Kasus Uji		Kasus Uji Validasi Perbarui Produk		
NO	**Test Case**	**Input**	**Expected**	**Result Test**
3	Member menginputkan judul kosong, deskripsi dengan nilai char, harga dengan nilai numerik, berat dengan nilai numerik, jumlah dengan nilai numerik,	Judul = null, deskripsi = kulkas dua pintu, harga = 1000, berat = 10, jumlah = 2	Gagal perbarui produk	Valid

6.2.9 Pengujian Validasi Bagikan Info Produk

Tabel 6.33 Kasus Uji Validasi Bagikan Info Produk

Nomor Kasus Uji Validasi		VAL_09		
Nama Kasus Uji		Kasus Uji Validasi Bagikan Info Produk		
NO	**Test Case**	**Input**	**Expected**	**Result Test**
1	Member menginputkan deskripsi dengan nilai string	Deskripsi = produkbaru	Berhasil Bagikan Info Produk	Valid
2	Member menginputkan deskripsi dengan nilai string	Deskripsi = 1232	Berhasil Bagikan Info Produk	Valid
3	Member tidak melakukan input pada deskripsi	Deskripsi = null	Berhasil Bagikan Info Produk	Valid

6.2.10 Pengujian Validasi Perbarui Profil Member

Tabel 6.34 Kasus Uji Validasi Perbarui Profile Member

Nomor Kasus Uji Validasi	**VAL_10**			
Nama Kasus Uji	**Kasus Uji Validasi Perbarui Profile Member**			
NO	**Test Case**	**Input**	**Expected**	**Result Test**
1	Member menginputkan nama dengan nilai string, email dengan nilai char, no handphone dengan nilai numerik dan catatan dengan nilai char	Nama = wildanmkf, email = wildan@mail.com, no handphone = 123455, catatan = @ini catatan	Berhasil Perbarui Profil Member	Valid
2	Member menginputkan nama dengan nilai string, email dengan nilai char, no handphone dengan nilai string dan catatan dengan nilai char	Nama = wildanmkf, email = wildan@mail.com, no handphone = duatiga, catatan = @ini catatan	Gagal Perbarui Profil Member	Valid
3	Member menginputkan nama kosong, email dengan nilai char, no handphone dengan nilai numerik dan catatan dengan nilai char	Nama = null, email = wildan@mail.com, no handphone = 123455, catatan = @ini catatan	Gagal Perbarui Profil Member	Valid

6.2.11 Pengujian Validasi Tambah item Keranjang Belanja

Tabel 6.35 Kasus Uji Validasi Tambah item Keranjang Belanja

Nomor Kasus Uji Validasi	**VAL_11**			
Nama Kasus Uji	**Kasus Uji Validasi Tambah item Keranjang Belanja**			
NO	**Test Case**	**Input**	**Expected**	**Result Test**
1	Member menginputkan jumlah barang dengan nilai numerik	Qty = 10	Berhasil tambah item keranjang belan	Valid

Nomor Kasus Uji Validasi		VAL_11		
Nama Kasus Uji		Kasus Uji Validasi Tambah item Keranjang Belanja		
NO	**Test Case**	**Input**	**Expected**	**Result Test**
2	Member menginputkan jumlah barang dengan nilai string	Qty = sepuluh	Gagal tambah item keranjang belan	Valid
3	Member menginputkan jumlah barang dengan nilai char	Qty = !1	Gagal tambah item keranjang belan	Valid

6.2.12 Pengujian Validasi Checkout

Tabel 6.36 Kasus Uji Validasi checkout

Nomor Kasus Uji Validasi		VAL_12		
Nama Kasus Uji		Kasus Uji Validasi checkout		
NO	**Test Case**		**Expected**	**Result Test**
1	Member menginputkan catatan dengan nilai string	Catatan = satu catatan	Berhasil Checkout	Valid
2	Member menginputkan catatan dengan nilai char	Catatan = satu@catatan	Berhasil Checkout	Valid
3	Member menginputkan catatan dengan nilai numerik	Catatan = 111	Berhasil Checkout	Valid

6.2.13 Pengujian Validasi Tambah Alamat

Tabel 6.37 Kasus Uji Validasi Tambah Alamat

Nomor Kasus Uji Validasi		VAL_13		
Nama Kasus Uji		Kasus Uji Validasi Tambah Alamat		
NO	**Test Case**		**Expected**	**Result Test**
1	Member menginputkan nama alamat dengan nilai string, kode pos dengan nilai numerik, jalan char	Nama alamat = alamatsatu, kodepos = 12323, jalan	Berhasil Tambah Alamat	Valid

Nomor Kasus Uji Validasi		VAL_13		
Nama Kasus Uji		Kasus Uji Validasi Tambah Alamat		
NO	**Test Case**		**Expected**	**Result Test**
		= jl. Lima belas		
2	Member menginputkan nama alamat kosong, kode pos dengan nilai numerik, jalan char	Nama alamat = null, kodepos = 12323, jalan = jl. Lima belas	Gagal Tambah Alamat	Valid
3	Member menginputkan nama alamat dengan nilai string, kode pos dengan nilai string, jalan char	Nama alamat = alamatsatu, kodepos = duabelas, jalan = jl. Lima belas	Gagal Tambah Alamat	Valid

6.2.14 Pengujian Validasi Tambah Testimoni

Tabel 6.38 Kasus Uji Validasi Tambah Testimoni

Nomor Kasus Uji Validasi		VAL_14		
Nama Kasus Uji		Kasus Uji Validasi Tambah Testimoni		
NO	**Test Case**	**Input**	**Expected**	**Result Test**
1	Member menginputkan testimoni dengan nilai string	Testimoni = produk bagus	Berhasil Tambah Testimoni	Valid
2	Member menginputkan testimoni dengan nilai char	Testimoni = pr0duk b@gus	Berhasil Tambah Testimoni	Valid
3	Member menginputkan testimoni kosong	Testimoni = null	Berhasil Tambah Testimoni	Valid

6.2.15 Pengujian Compatibility

Pada pengujian compatibility ini berfungsi untuk menguji kebutuhan non fungsional. Berikut proses pengujian kebutuhan non fungsinal yang telah dilakukan.

Tabel 6.39 Kasus Uji Menjalankan Fungsionalitas Sistem Pada Browser Yang Berbeda

Nomor Kasus Uji Validasi	**VAL_15**
Nama Kasus Uji	**Kasus uji menjalankan fungsinalitas sistem pada browser yang berbeda**
Prosedur	**Expected**
Menjalankan sebagian fungsionalitas yang ada pada browser Google Chrome, Mozilla Firefox dan Microsoft Edge Fungsionalitas yang dijalankan meliputi: 1. Masuk admin 2. Tambah data produk kategori 3. Perbarui data produk kategori 4. Tambah user admin 5. Daftar 6. Masuk 7. Tambah produk 8. Perbarui produk 9. Bagikan info produk 10. Perbarui profil member 11. Tambah item keranjang belanja 12. Checkout 13. Tambah alamat 14. Tambah testimoni	Sebagian dari funsionalitas sistem yang dijalankan dapat berjalan secara normal.

Pada tabel 6.32 merupakan hasil dari pengujian *compatibility* yang telah dilakukan. Berikut hasilnya:

Tabel 6.40 Hasil Pengujian Compatibility

Nomor Kasus uji validasi	Hasil Pengujian		
	Google Chrome	Mozilla Firefox	Microsoft Edge
VAL_1	*Valid*	*Valid*	*Valid*
VAL_2	*Valid*	*Valid*	*Valid*
VAL_3	*Valid*	*Valid*	*Valid*
VAL_4	*Valid*	*Valid*	*Valid*
VAL_5	*Valid*	*Valid*	*Valid*
VAL_6	*Valid*	*Valid*	*Valid*
VAL_7	*Valid*	*Valid*	*Valid*
VAL_8	*Valid*	*Valid*	*Valid*
VAL_9	*Valid*	*Valid*	*Valid*
VAL_11	*Valid*	*Valid*	*Valid*
VAL_12	*Valid*	*Valid*	*Valid*
VAL_13	*Valid*	*Valid*	*Valid*
VAL_14	*Valid*	*Valid*	*Valid*

6.3 Pengujian Kebutuhan

Pengujian kebutuhan merupakan pengujian yang melibatkan kasus uji pada setiap spesifikasi kebutuhan yang berhubungan dengan sistem. Untuk melakukan pengujian kebutuhan pada setiap spesifikasi kebutuhan bisa ditelusi dengan kasus uji menggunakan *traceability matrix.*

6.3.1 Traceability Matrix

Traceability matrix atau *Requirement Traceability Matrix - RTM* adalah tabel yang berisi daftar requirements, artibut setiap requirement, dan status dari requirement yang digunakan untuk memastikan pada semua spesifikasi kebutuhan telah terpenuhi dengan baik.

Tabel 6.41 Traceability Matrix

No	Test Case ID	No Use Case Skenario	Functional Requirement ID	Functional Requirement Name	Validasi Test ID	Result Test
1	TC_01	1	SM_F_01	Masuk admin	VAL_01	*Valid*

2	TC_07	7	SM_F_07	Tambah data produk kategori	VAL_02	*Valid*
3	TC_08	8	SM_F_08	Perbarui data produk kategori	VAL_03	*Valid*
4	TC_12	12	SM_F_12	Tambah user admin	VAL_04	*Valid*
5	TC_14	14	SM_F_14	Daftar	VAL_05	*Valid*
6	TC_15	15	SM_F_15	Masuk	VAL_06	*Valid*
7	TC_20	20	SM_F_20	Tambah produk	VAL_07	*Valid*
8	TC_21	21	SM_F_21	Perbarui produk	VAL_08	*Valid*
9	TC_24	24	SM_F_24	Bagikan info produk	VAL_09	*Valid*
10	TC_29	29	SM_F_29	Perbarui profil member	VAL_10	*Valid*
11	TC_30	30	SM_F_30	Tambah item keranjang belanja	VAL_11	*Valid*
12	TC_37	37	SM_F_37	Checkout	VAL_12	*Valid*
13	TC_39	39	SM_F_39	Tambah alamat member	VAL_13	*Valid*
14	TC_41	41	SM_F_41	Tambah testimoni	VAL_14	*Valid*

6.4 Analisis Pengujian

Pada analisis pengujian ini bertujuan untuk mendapatkan hasil kesimpulan dari pengujian sistem *marketplace* yang dapat merekomendasikan *grup facebook* yang sesuai dengan produk menggunakan algoritme *cosine similarity*. Untuk mendapatkan hasil analisis maka memerlukan proses yang meliputi pengujian unit dan pengujian validasi. Pengujian unit merupakan suatu pengujian yang didapat dari hasil pengujian algoritme kode dengan menggunakan teknik *white-box* dengan teknik pengujian *basis-path*. Sedangkan pengujian validasi merupakan

yang dilakukan dengan cara memantau kinerja sistem dengan daftar kebutuhan fungsional ataupun non fungsional.

Hasil dari pengujian unit yang telah diimplementasikan mendapatkan hasil tiga jalur dari pengujian unit klas koneksifb operasi index, , lima jalur dari pengujian unit klas Cosinesimilarity operasi perhitungan(), dua jalur dari pengujian unit klas Cosinesimilarity operasi postfacebook (), dua jalur dari pengujian unit klas KeranjangBelanja operasi addcart(), dua jalur dari pengujian unit klas KeranjangBelanja operasi checkout(), tiga jalur dari pengujian unit klas KeranjangBelanja operasi getInvoice, dua jalur dari pengujian unit DataTransaksi operasi update() dan empat jalur dari pengujian unit Transaksi operasi update.

Sedangkan hasil dari pengujian fungsionalitas dengan black-box testing pada 14 pengujian dengan menggunakan metode Equivalence Partitioning hasilnya 100%(valid) dengan demikian hasil pengujian fungsionalitas dapat dikatakan sangat baik.

Dari hasil pengujian diatas dapat disimpulkan bahwa implementasi dan fungsional dari sistem yang dibangun dapat berjalan sesuai dengan spesifikasi kebutuhan yang telah dijabarkan pada analisis kebutuhan yang terdapat pada pembahasan.

BAB 7 PENUTUP

7.1 Kesimpulan

Bedasarkan hasil perancangan, implementasi dan pengujian yang dilakukan, maka dapat diambil kesimpulan sebagai berikut:

1. Pada analisis kebutuhan sistem *marketplace* menghasilkan empat aktor yaitu admin, pembeli, penjual dan user kemudian menghasilkan 41 kebutuhan fungsional, untuk kebutuhan non fungsional menghasilkan satu kebutuhan yaitu *compatibility* dimana sistem *marketplace* harus dapat digunakan pada *browser chrome, firefox* dan *microsoft edge.*
2. Pada tahap perancangan yang telah dilakukan didapatkan hasil *sequence diagram*, *class diagram*, perancangan algoritme dan perancangan antar muka. Sistem *marketplace* memiliki fitur-fitur utama sesui dengan analisis kebutuhan yaitu dapat melakukan melakukan *posting* produk yang dijual, dapat melakukan transaksi jual beli yang langsung dikonfirmasi oleh *admin* sistem *marketplace* dan kemudian transaksi diteruskan ke penjual produk untuk diproses selanjutnya, dapat melihat seberapa mirip antara produk dengan grup *facebook* yang tersedia dan juga dapat membagikan informasi produk ke *grup facebook* sesui dengan yang dinginkan oleh penjual.
3. Pada tahap pengujian dibagi menjadi dua bagian yaitu pengujian fungsional dan pengujian non-fungsional. Pengujian fungsional dilakukan dengan mengunakan teknik *white-box* dan *black-box.* Pengujian *white-box* dilakukan dengan cara pengujian unit . Pengujian unit ini dilakukan pada delapan operasi yang berbeda semuanya mendapatkan hasil 100% *valid* dan sesuai dengan analisis kebutuhan yang telah dibuat. Pada pengujian *black-box* dilakukan pada 14 kebutuhan fungsional, pada pengujian ini menggunakan metode *Equivalence Partitioning*. Dan dari pengujian *black-box* ini menghasilkan 14 kebutuhan yang dapat dinyatakan 100% *valid*. Untuk kebutuhan non-fungsional pengujian dilakukan dengan pengujian *compatibility*, dan dapat disimpulkan bahwa sistem *marketplace* yang dapat merekomendasikan *grup facebook* yang sesuai dengan produk menggunakan algoritme *cosine similarity* berjalan dengan baik pada *browser Google Chrome, Mozilla Firefox dan Microsoft Edge.*

7.2 Saran

Adapun saran yang dapat diberikan untuk kegiatan pengembangan pembangunan sistem *marketplace* yang dapat merekomendasikan grup *facebook* yang sesuai dengan produk menggunakan algoritme cosine similarity selanjutnya adalah sebagai berikut :

1. Menambahkan lebih banyak terhubung ke sosial media lainya agar informasi produk lebih banyak tersebar.

2. Untuk pengembangan lebih lanjut dapat dibuat aplikasi perangkat *mobile* (android / *IOS device*).
3. Untuk pengembangan lebih lanjut dapat menambahkan fitur Graph API marketing agar member bisa langsung memasang iklan pada facebook melalui sistem *marketplace* yang telah dibuat tanpa harus masuk kehalaman facebook terlebih dahulu.
4. Pada pengembangan selanjutnya diharapkan pembuatan *user interface* yang lebih menarik dan mudah digunakan.

DAFTAR PUSTAKA

Ariantini Dewa A R.2016. *Pengukuran Kemiripan Dokumen Teks Bahasa Indonesia Menggunakan Metode Cosine Similarity.* E-Journal Teknik Informatika Volume 9, No 1 (2016)

Arifin Muhammad. 2015. *Analisis Dan Perancangan Sistem Informasi Praktek Krja Lapangan Pada Instansi/Perusahaan*. Jurnal Simetris

Ekawati Ni Wayan. 2012. *Jejaring Sosial/Facebook Sebagai Media E-Pengecer.* Buletin Studi Ekonomi, Volume 17, No. 2, Agustus 2012

Enricko Lukman. 2013. Laporan: inilah yang dilakukan 74,6 juta pengguna internet Indonesia ketika online. [online] Tersedia di: <https://id.techinasia.com/ tingkah-laku-pengguna-internet-indonesia> [Diakses 5 April 2017]

Fadul, F. (2014). Apa Itu Bootstrap? *Bagaimana Memulai Belajar Bootstrap Untuk Pemula*. Retrieved from dul.web.id: http://dul.web.id/bootstrap/3/tuts-tips/belajar-bootstrap-untuk-pemula.php [Diakses 10 Maret 2017]

Haviludin. 2011. *Memahami Penggunaan UML (Unified Modelling Language)*. Jurnal Informatika Mulawarman Vol 6 No. 1 Febuari 2011 1

Herwijayanti Bening. 2017. *Klasifikasi Berita Online dengan menggunakan Pembobotan TF-IDF dan Cosine Similarity.* Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer Vol. 2, No. 1, Januari 2018, hlm. 306-312 e-ISSN: 2548-964X

ISO 9241-11: 1998, *Ergonomic reqirements for office work with visual display terminals (VDTs)* – Part 11: Guidance on usability

Kaur Jasmeet, Singh Neha. 2014. *Facebook Integration with RESTFB API.* International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 3 Issue 11, November 2014

Limpo Utami Ervina, Wibowo Adi, Lim Resmania. 2012. Pembuatan Jaringan Sosial Peneliti berbasi Facebook yang memanfaatkan Situs Artikel Ilmiah di Pusat penelitian Universitas Kristen Petra.

Mansur. 2015. *Business To Business (B2b) E-Marketplace Sebagai Media Promosi Produk Usaha Kecil Dan Menengah (Ukm).* Jurnal Politeknik Negeri Bengkalis Volume 01, No. 01, Februari 2015 No. ISSN: 2442-885X

Muningsih Elly. 2014. *Facebook Commerce, E-Commerce Pada Media Sosial Facebook Yang Modern Dan Populer.* Bianglala Informatika Vol . II No 1 Maret 2014

Pressman, R.S. 2015. Rekayasa Perangkat Lunak: Pendekatan Praktisi Buku I. Yogyakarta: Andi

Pressman, R.S. 2010, Software Engineering : a practitioner's approach, McGrawHill, New York

Putra Angga Kurnia, Nyoto Rudy dwi dan Pratiwi Helen Sasty. 2017. *Rancang bangun Aplikasi Marketplace Penyedia Jasa Les Private DI Kota Pontianak berbasis web*. Jurnal Sistem dan Teknologi Informasi (JUSTIN) Vol. 2, No. 1.

Sadgotra Wisnu yoga. 2013. *Perancangan Online Marketplace Untuk Usaha kecil dan menengah (UKM) di Kabupaten Purworejo*. Jurnal imiah DASI

Shi Mingtao, 2010. Software Functional Testing from the Perspective of Business Practice Computer and Information Science. Vol. 3, No. 4; November 2010 . www.ccssenet.org/cis

Sidi M. Mustaqbal, *Pengujian Aplikasi Menggunakan Black Box Testing Boundary Value Analysis.* Jurnal Ilmiah Teknologi Informasi Terapan Volume I, No 3, 10 Agustus 2015

Widiyanto Ibnu, Prasilowati Sri lestari. 2015 . *Perilaku pembelian Melalui Internet*. JMK, VOL 17 No. 2 ISSN

Zakir Ahmad. 2016. *Rancang Bangun Responsive Web Layout Dengan Menggunakan Bootstrap Framework.* InfoTekJar (Jurnal Nasional Informatika dan Teknologi Jaringan) Vol 1, No 1, September 2016